Atsuko Miyaji
Hiroaki Kikuchi
Kai Rannenberg (Eds.)

LNCS 4752

Advances in Information and Computer Security

Second International Workshop on Security, IWSEC 2007
Nara, Japan, October 2007
Proceedings

Springer

Lecture Notes in Computer Science 4752

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Atsuko Miyaji Hiroaki Kikuchi
Kai Rannenberg (Eds.)

Advances in Information and Computer Security

Second International Workshop on Security, IWSEC 2007
Nara, Japan, October 29-31, 2007
Proceedings

Springer

Volume Editors

Atsuko Miyaji
Japan Advanced Institute of Science and Technology
School of Information Science
Ishikawa, Japan
E-mail: miyaji@jaist.ac.jp

Hiroaki Kikuchi
Tokai University
School of Information Technology and Electronics
Kanagawa, Japan
E-mail: kikn@tokai.ac.jp

Kai Rannenberg
Goethe University Frankfurt
Institute of Business Informatics
Frankfurt/Main, Germany
E-mail: kai.rannenberg@m-lehrstuhl.de

Library of Congress Control Number: Applied for

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, K.4.1, C.2, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-540-75650-7 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-75650-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2007
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12041589 06/3180 5 4 3 2 1 0

Preface

The International Workshop on Security (IWSEC 2007) was the second in the annual series that started in 2006. IWSEC 2007 was held at the New Public Hall in Nara, Japan, during October 29–31, 2007.

This year there were 112 paper submissions, and from these 30 papers were accepted. Accepted papers came from 27 different countries, with the largest proportion coming from Japan (12). Estonia, China, Korea, Spain, Taiwan and the USA contributed 2 papers each and Canada, Germany, Greece, Poland, Turkey and Vietnam contributed 1 paper each. We would like to thank all of the authors who submitted papers to IWSEC 2007.

The contributed papers were supplemented by one invited talk from the eminent researcher Prof. Doug Tygar (UC Berkeley) in information security.

We were fortunate to have an energetic team of experts who formed the Program Committee. Their names may be found overleaf, and we are sincerely grateful for all their great efforts. This team was supported by an even larger number of individuals who reviewed papers in their particular areas of expertise. A list of these names is also provided; we hope it is complete.

We are delighted to acknowledge the generous financial sponsorship of IWSEC 2007 by Carnegie Mellon CyLab Japan, the International Communication Foundation (ICF), and the National Institute of Information and Communications Technology (NICT). The workshop was co-sponsored jointly by ISEC, a technical group on information security of IEICE (The Institute of Electronics, Information and Communication Engineers) and CSEC, a special interest group on computer security of IPSJ (Information Processing Society of Japan). The excellent Local Organizing Committee was led by the IWSEC 2007 General Chairs, Prof. Masakatu Morii and Dr. Masato Terada.

October 2007

Atsuko Miyaji
Hiroaki Kikuchi
Kai Rannenberg

IWSEC 2007
Second International Workshop on Security

[illegible]

ISEC (Technical Group on Information Security, Engineering Sciences Society,
of the Institute of Electronics, Information and Communication Engineers,
Japan)
CSEC (Special Interest Group on Computer Security of Information Processing
Society of Japan)

[illegible]

Carnegie Mellon CyLab Japan
International Communication Foundation (ICF)
National Institute of Information and Communications Technology (NICT)

General Co-chairs

Masakatu Morii (Kobe University, Japan)
Masato Terada (Hitachi Ltd., Japan)

Program Committee Co-chairs

Atsuko Miyaji (Japan Advanced Institute of Science and Technology, Japan)
Hiroaki Kikuchi (Tokai University, Japan)
Kai Rannenberg (Goethe University Frankfurt, Germany)

Advisory Committee

Norihisa Doi (Chuo University, Japan)
Akira Hayashi (Kanazawa Institute of Technology, Japan)
Hideki Imai (Chuo University, Japan)
Guenter Mueller (University of Freiburg, Germany)
Yuko Murayama (Iwate Prefectural University, Japan)
Eiji Okamoto (University of Tsukuba, Japan)
Ryoichi Sasaki (Tokyo Denki University, Japan)
Shigeo Tsujii (Institute of Information Security, Japan)

Local Organizing Committee

Finance Chairs	Yoshiaki Shiraishi (Nagoya Institute of Technology, Japan) Masayuki Terada (NTT DoCoMo, Inc., Japan) Keisuke Takemori (KDDI R&D Laboratories Inc., Japan)
Publicity Chairs	Naoto Sone (Naruto University of Education, Japan) Tsuyoshi Takagi (Future University Hakodate, Japan) Koji Chida (NTT Co., Japan)
Local Arrangement Chairs	Hidenori Kuwakado (Kobe University, Japan) Masakatsu Nishigaki (Shizuoka University, Japan) Yuji Suga (Canon Inc., Japan)
Publication Chairs	Toshihiro Tabata (Okayama University, Japan) Masahiro Mambo (University of Tsukuba, Japan) Isao Echizen (National Institute of Informatics, Japan)
Registration Chairs	Mitsuru Matsui (Mitsubishi Electric Co., Japan) Masayuki Terada (NTT DoCoMo, Inc., Japan) Keisuke Takemori (KDDI R&D Laboratories Inc., Japan)
Award Chairs	Hiroshi Doi (Institute of Information Security, Japan) Hiroshi Yoshiura (University of Electro-Communications, Japan) Tetsuya Izu (Fujitsu Laboratories Ltd., Japan)
Liaison Chairs	Katsunari Yoshioka (National Institute of Information and Communications Technology, Japan) Takehisa Kato (Toshiba Solutions Corporation, Japan) Ryuya Uda (Tokyo University of Technology, Japan)

Program Committee

Koichiro Akiyama (Toshiba Corporation, Japan)
Tomoyuki Asano (Sony Corporation, Japan)
Feng Bao (Institute for Infocomm Research, Singapore)
Kevin Butler (Pennsylvania State University, USA)
George Robert Blakley, Jr. (Texas A&M University, USA)
Liqun Chen (HP Laboratories, UK)
Soon Ae Chun (City University of New York, USA)

George Danezis (Katholieke Universiteit Leuven, Belgium)
Ed Dawson (Queensland University of Technology, Australia)
Bart De Decker (Katholieke Universiteit Leuven, Belgium)
Hiroshi Doi (Institute of Information Security, Japan)
Ee-Chien Chang (National University of Singapore, Singapore)
Steven Furnell (University of Plymouth, UK)
Soichi Furuya (Hitachi, Ltd., Japan)
Dimitris Gritzalis (Athens University of Economics & Business, Greece)
Shoichi Hirose (University of Fukui, Japan)
Keiichi Iwamura (Tokyo University of Science, Japan)
Tetsu Iwata (Nagoya University, Japan)
Marc Joye (Thomson R&D, France)
Angelos D. Keromytis (Columbia University, USA)
Kwangjo Kim (Information and Communications University, Korea)
Michiharu Kudo (IBM Japan, Japan)
Noboru Kunihiro (The University of Electro-Communications, Japan)
Xuejia Lai (Shanghai Jiao Tong University, China)
Kwok-Yan Lam (Tsinghua University, China)
Byoungcheon Lee (Joongbu University, Korea)
Dong Hoon Lee (Korea University, Korea)
Pil Joong Lee (Pohang University of Science and Technology, Korea)
William List (Firm Wm. List & Co, UK)
Javier Lopez (University of Malaga, Spain)
Masahiro Mambo (University of Tsukuba, Japan)
Mark Manulis (Ruhr-University of Bochum, Germany)
Wenbo Mao (HP Labs, China)
Keith Martin (Royal Holloway, University of London, UK)
Mitsuru Matsui (Mitsubishi Electric, Japan)
Kanta Matsuura (University of Tokyo, Japan)
Toru Nakanishi (Okayama University, Japan)
Masakatsu Nishigaki (Shizuoka University, Japan)
Takeshi Okamoto (University of Tsukuba, Japan)
Kazumasa Omote (Fujitsu Laboratories, Ltd., Japan)
Kenneth G. Paterson (Royal Holloway, University of London, UK)
Adrian Perrig (Carnegie Mellon University, USA)
Raphael Phan (EPFL, Switzerland)
Rei Safavi-Naini (University of Wollongong, Australia)
Kouichi Sakurai (Kyushu University, Japan)
Masakazu Soshi (Japan Advanced Institute of Science and Technology, Japan)
Willy Susilo (University of Wollongong, Australia)
Mitsuru Tada (Chiba University, Japan)
Tsuyoshi Takagi (Future University Hakodate, Japan)
Toshiaki Tanaka (KDDI R&D Laboratories Inc., Japan)
Ryuya Uda (Tokyo University of Technology, Japan)
Serge Vaudenay (EPFL, Switzerland)

Jozef Vyskoc (VaF, Slovak Republic)
Hajime Watanabe (National Institute of Advanced Science and Technology, Japan)
Duncan Wong (City University of Hong Kong, China)
Sung-Ming Yen (National Central University, Taiwan)
Hiroshi Yoshiura (University of Electro-Communications, Japan)
Yuliang Zheng (University of North Carolina, USA)
Jianying Zhou (Institute for Infocomm Research, Singapore)
Alf Zugenmaier (DoCoMo Euro-Labs, Germany)

External Reviewers

Toru Akishita, Elli Androulaki, Michael Arnold, Man Ho Au, Jean-Philippe Aumasson, Thomas Baignères, Tor E. Bjorstad, Jeffrey Bloom, Emmanuel Bresson, Matt Burnside, Haibo Chen, Kuo-Zhe Chiou, Kim-Kwang Raymond Choo, Siu-Leung Chung, Carlos Cid, Lizzie Coles-Kemp, Debra L. Cook, Jason Crampton, Gabriela Cretu, Jintai Ding, Ling Dong, Stelios Dritsas, Dang Nguyen Duc, William Enck, Koichi Fujisaki, Kazuhide Fukushima, Soichi Furuya, Steven Galbraith, Meng Ge, Willi Geiselmann, Chris Grier, Tim Güneysu, Satoshi Hada, Yu Haifeng, Goichiro Hanaoka, Yoshikazu Hanatani, Yoshiki Higashikado, Jin Ho Kim, Katrin Hoeper, Dennis Hofheinz, Yoshiaki Hori, Chao-Chih Hsu, Qiong Huang, Taichi Isogai, Yukio Izumi, Ik Rae Jeong, Shaoquan Jiang, Mohamed Karroumi, Yasuharu Katsuno, Young Mok Kim, Samuel T. King, Wataru Kitada, Divyan M. Konidala, Masafumi Kusakawa, Eun Jeong Kwon, Frédéric Lefèbfre, Arjen K. Lenstra, Wanqing Li, Vo Duc Liem, Wei-Chih Lien, Hoon Wei Lim, Hsi-Chung Lin, Yu Long, Antoine Monsifrot, Shiho Moriai, Cedric Ng, Steven Noel, Satoshi Obana, Yutaka Oiwa, Hiroyuki Okazaki, Haruki Ota, Toru Owada, Je Hong Park, Sylvain Pasini, Francesco De Pellegrini, Ludovic Perret, Rodrigo Roman, Eun-Kyung Ryu, Jae Woo Seo, Joshua Schiffman, Jörg Schwenk, Mike Scott, Abhi Shelat, Nicholas Sheppard, Jong Hoon Shin, Masaaki Shirase, Stelios Sidiroglou, Leonie Simpson, Yingbo Song, Rainer Steinwandt, Chunhua Su, Hongwei Sun, Daisuke Suzuki, Koutarou Suzuki, Amril Syalim, Gelareh Taban, Kenichi Takahashi, Takeaki Terada, Marianthi Theoharidou, Xiaojian Tian, Patrick Traynor, Eran Tromer, Bill Tsoumas, Jheng-Hong Tu, Yoshifumi Ueshige, Kentaro Umesawa, Kristof Verslype, Jose L. Vivas, Yuji Watanabe, Nicholas Weaver, Yongdong Wu, Shinji Yamanaka, Guomin Yang, Yeon-Hyeong Yang, Tomoko Yonemura, Maki Yoshida, Rui Zhang, Huafei Zhu

Table of Contents

Software and Multimedia Security

Public-Key Cryptography (1)

Network Security

E-commerce and Voting

Operating Systems

Public-Key Cryptography (2)

Security and Information Management

Anonymity and Privacy

Digital Signatures, Hash Function and Protocol

A Note on the (Im)possibility of Using Obfuscators to Transform Private-Key Encryption into Public-Key Encryption

Satoshi Hada[1] and Kouichi Sakurai[2]

[1] Tokyo Research Laboratory, IBM Research, 1623-14, Shimotsuruma, Yamato,
Kanagawa 242-8502, Japan
satoshih@jp.ibm.com

[2] Dept. of Computer Science and Communication Engineering, Kyushu University,
744 Motooka, Nishi-ku, Fukuoka, Fukuoka 819-0395, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. Transforming private-key encryption schemes into public-key encryption schemes is an interesting application of program obfuscation. The idea is that, given a private-key encryption scheme, an obfuscation of an encryption program with a private key embedded is used as a public key and the private key is used for decryption as it is. The security of the resulting public-key encryption scheme would be ensured because obfuscation is *unintelligible* and the public key is expected to leak no information on the private key. This paper investigates the possibility of general-purpose obfuscators for such a transformation, i.e., obfuscators that can transform an arbitrary private-key encryption scheme into a secure public-key encryption scheme. Barak et al. have shown a negative result, which says that there is a *deterministic* private-key encryption scheme that is *unobfuscatable* in the sense that, given any encryption program with a private key embedded, one can efficiently compute the private key. However, it is an open problem whether their result extends to *probabilistic* encryption schemes, where we should consider a relaxed notion of obfuscators, i.e., *sampling obfuscators*. Programs obfuscated by sampling obfuscators do not necessarily compute the same function as the original program, but produce the same distribution as the original program. In this paper, we show that there is a *probabilistic* private-key encryption scheme that can not be transformed into a secure public-key encryption scheme by sampling obfuscators which have a special property regarding input-output dependency of encryption programs. Our intention is not to claim that the required special property is reasonable. Rather, we claim that general-purpose obfuscators for the transformation, if they exist, must be a sampling obfuscator which does NOT have the special property.

1 Introduction

1.1 Obfuscation

An obfuscator is a tool to convert a program into a new program which is [illegible] while preserving the functionality. Several formal definitions have

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 1–12, 2007.
© Springer-Verlag Berlin Heidelberg 2007

been proposed so far [13,1,17,18,10,14]. Informally, obfuscators should satisfy the following two requirements: (1) functionality: the new program has the same functionality as the original one and (2) virtual black-box property: whatever one can efficiently compute given the new program can be computed given oracle access to the functionality. The functionality requirement is a syntactic requirement while the virtual black-box property represents the security requirement that the obfuscated program should be unintelligible.

As discussed in [1], obfuscators, if they exist, would have a wide variety of cryptographic applications including software protection, homomorphic encryption, removing random oracles, and transforming private-key encryption schemes into public-key encryption schemes. Unfortunately, the impossibility of generic obfuscation have been shown in [1,10] (even under very weak definitions based on the virtual black-box property). For example, as shown in [1], there exists a family of functions $\mathcal{F}$ that are [illegible] in the sense that there is a boolean property of functions such that, given any program that computes a function $f \in \mathcal{F}$, the property of f can be efficiently computed, yet given oracle access to a randomly selected function $f \in \mathcal{F}$, no efficient algorithm can compute the property of f much better than random guessing. However, such negative results do not rule out the possibility that there exists an obfuscator for a [illegible] set of programs (a specific application). Indeed, some positive results are known for point functions [2,3,17,18,10,6,14] and re-encryption [15].

When we consider obfuscation of [illegible] algorithms (such as probabilistic encryption algorithms), we must be careful; There are two different definitions of the functionality requirement. We recall them informally in terms of obfuscation of probabilistic encryption algorithms. Let $\mathcal{E}_K(M, R)$ be a private-key encryption program, where K is an embedded private key, M is a plaintext, and R is a set of random coins. Similarly, let $\mathcal{E}'_K(M, R)$ be an obfuscation of it. The first definition is the usual one and requires that the two programs compute the same function, i.e., for every pair (M, R), we have $\mathcal{E}_K(M, R) = \mathcal{E}'_K(M, R)$. In this paper, obfuscators satisfying this functionality requirement are called [illegible] [illegible][1]. On the other hand, the second definition requires that, for every M, the two distributions obtained by evaluating $\mathcal{E}_K(M, R)$ and $\mathcal{E}'_K(M, R)$ on independent random coins R are the same. This is a relaxed requirement, but it would be sufficient for cryptographic applications as noted in [14,15]. We call obfuscators satisfying this functionality requirement as [illegible] as in [1, Section 6].

1.2 Transforming Private-Key Encryption into Public-Key Encryption

Transforming private-key encryption schemes into public-key encryption schemes is an interesting application of obfuscation. The idea is that, given a private-key encryption scheme, an obfuscation of an encryption program with a private key embedded is used as a public key and the private key is used for decryption as

[1] In this paper, programs are defined by boolean circuits.

it is. Let $\mathcal{E}_K(M, R)$ be a probabilistic private-key encryption program, where K is an embedded private key, M is a plaintext, and R is a set of random coins. Then we obfuscate it into a new encryption program $\mathcal{E}'_K(M, R)$, which we use as the public key. When we want to encrypt a message M by the public key, we pick a set of random coins R and execute the public key, i.e., the obfuscated program $\mathcal{E}'_K$ on (M, R). We expect that the public key reveals no information on the private key because the obfuscated program is unintelligible. In this sense, the resulting public key encryption scheme could satisfy some sort of security requirement. As mentioned above, the generic impossibility results of [1,10] does not rule out the possibility that we have a [illegible] obfuscator for such a transformation. By "general-purpose obfuscators", we mean obfuscators that can transform an arbitrary private-key encryption scheme into a secure public-key encryption scheme.

The transformation is very interesting for at least two reasons.

1. Impagliazzo and Rudich showed that there exists no black-box reduction from private-key encryption schemes into public-key encryption schemes [16]. The transformation by an obfuscator can bypass their impossibility result.
2. It was an original idea suggested by Diffie and Hellman in their seminal paper [5] to design a public-key encryption scheme (Recall that, when the paper was published, there was no candidate public-key encryption scheme). So we can say that it is a natural principle for the design of public-key encryption schemes. We may be able to construct a (totally) new public-key encryption scheme using this idea.

It is important to note that we should consider [illegible] private-key encryption schemes for this transformation to make sense. When we transform a [illegible] private-key encryption scheme, the resulting candidate public-key encryption scheme is deterministic as well. No deterministic public-key encryption scheme is secure in the usual sense [11].

Hofheinz et al. provided a formal treatment of the transformation under their proposed definitions of the virtual black-box property [14]. They showed that a probabilistic private-key encryption scheme secure against chosen-plaintext attacks can be transformed into a probabilistic public-key encryption scheme secure against chosen-plaintext attacks if an obfuscator for the private-key scheme exists according to their definitions.

1.3 Motivating Question

Our motivating question is: Does such a general-purpose obfuscator exist? We already have at least two negative answers to this question. Both answers are based on the existence of private-key encryption schemes that are "unobfuscatable" in some sense. We need to be careful because the meaning of "unobfuscatable" is different.

The first answer is by [1, Section 4.3]. They constructed a [illegible] private-key encryption scheme that is [illegible] in the sense that, given any encryption program with a private key embedded, one can efficiently compute the private key (but it is hard to do so given oracle access to the encryption function). This means that no matter how we define the security (the virtual black-box property) of obfuscators, the resulting public-key encryption scheme is totally insecure. However, their argument does not extend to obfuscation of [illegible] private-key encryption schemes by [illegible].

The second answer is by [14]. They showed that there exist [illegible] private-key encryption schemes that are [illegible] in the sense that they can not be obfuscated under their specific definitions of the virtual black-box property[2]. Therefore, the meaning of "unobfuscatable" is different from the above. Their answer would be less negative than the first one in the sense that we may be able to relax the virtual black-box property so that the encryption schemes become obfuscatable and the resulting public-key encryption schemes still satisfy some meaningful security. However, it would be stronger for two reasons: (1) it deals with [illegible] encryption and (2) it deals with [illegible] while the first answer deals with circuit obfuscators.

In summary, it is an open problem whether there exists a [illegible] private-key encryption scheme that is [illegible] in the sense of the first answer.

1.4 Our Contribution

In this paper, we will give some observations on the above open problem by constructing [illegible] private-key encryption schemes that no obfuscator can transform into a secure public-key encryption scheme. Our approach is different from the approaches taken by [1, Section 4.3] and [14]. The both papers focused on the impossibility of obfuscating encryption programs, which in turn implies that the transformation does not work. In this case, using the term "unobfuscatable" makes sense. On the other hand, our approach will focus on the impossibility of the transformation itself directly. Basically, our results say that there exists a probabilistic private-key encryption scheme that can not be transformed into a secure public-key encryption scheme [illegible]. Therefore, we don't say that the private-key encryption schemes that we will construct are "unobfuscatable." Instead, we say that they are "untransformable." Readers may think that our results could contradict the positive result of [14], which says that a probabilistic private-key encryption scheme secure against chosen-plaintext attacks can be transformed into a probabilistic public-key encryption scheme secure against chosen-plaintext attacks if an obfuscator satisfying a virtual black-box property exists for it. However, there is no contradiction because the virtual black-box property defined in [14] is different from the one discussed in this paper (See Section 3.3 for more details).

[2] More specifically, any scheme that enjoys *ciphertext integrity* in a publicly verifiable way is *unobfuscatable* in the sense that the virtual black-box property of their proposed definition requires that a simulator can produce an obfuscated signing key given oracle access to the encryption oracle, but it would be hard due to the unforgeability of the underlying signature scheme. See [14] for more details.

First, we will show that there exists a [illegible] private-key encryption scheme that is [illegible] by circuit obfuscators. It is untransformable in the sense that, given oracle access to any encryption program with a private key embedded, it is easy to decrypt any ciphertext (even if it could be hard to recover the private key). It is the standard encryption scheme based on pseudorandom functions.

$$\mathcal{E}_K(M,R) = (C_1 = M \oplus f_K(R), C_2 = R)$$

where f_K is a pseudorandom function. Note that the ciphertext (C_1, C_2) can be decrypted using oracle access to the encryption algorithm, i.e., $\mathcal{E}_K(C_1, C_2)$ outputs (M, C_2). Therefore, no matter how we obfuscate the encryption program, we can use the obfuscated program as a black-box to decrypt every ciphertext and the transformation does not work at all.

The next question is whether there exists a probabilistic private-key encryption scheme that is untransformable [illegible]. This time, the obfuscated program $\mathcal{E}'_K(M,R)$ does not necessarily compute the same function as $\mathcal{E}_K(M,R)$ and so there is no guarantee that $\mathcal{E}'_K(M,R)$ can be used for decryption. We will construct a probabilistic encryption scheme that is untransformable even by sampling obfuscators by modifying the above standard scheme as follows:

$$\mathcal{E}_K(M,(R_1,R_2)) = (M \oplus f_K(R_1), R_1, f_K(R_2 \oplus M), R_2)$$

We will show that, given oracle access to any obfuscated encryption program that samples the same distribution, we can efficiently compute the pseudorandom function f_K, which enables us to decrypt every ciphertext. However, we require that sampling obfuscators satisfy a special property, that is, preserve an input-output dependency of programs (to be defined formally in Section 3). Therefore, we cannot say that we completely solve the open problem.

We stress that our intention is NOT to claim that the required special property is reasonable. Our emphasis is on its implication. That is, general-purpose obfuscators for the transformation from private-key encryption into public-key encryption, if they exist, must be a sampling obfuscator which does NOT preserve the input-output dependency.

[illegible] As it is clear from the above description, we allow adversaries to set the random coins when they execute a given public key (an obfuscated encryption program). This ability for adversaries to set the random coins is inherent with the transformation.

2 Preliminaries

We say that a function $\nu(\cdot) : \mathbb{N} \to \mathbb{R}$ is negligible in n if for every polynomial $p(\cdot)$ and all sufficiently large n's, it holds that $\nu(n) < 1/p(n)$. PPT stands for "probabilistic polynomial time".

Given a probability distribution S, we denote by $x \leftarrow S$ the operation of selecting an element according to S. If A is a probabilistic machine then $A(x_1, x_2, \ldots, x_k)$ denotes the output distribution of A on inputs $(x_1, x_2, \ldots, x_k)$. Let $\Pr[x \leftarrow S_1; x_2 \leftarrow S_2; \ldots; x_k \leftarrow S_k : E]$ denote the probability of the event E after the processes $x_1 \leftarrow S_1, x_2 \leftarrow S_2, \ldots, x_k \leftarrow S_k$ are performed in order.

Definition 1. (l_{in}, l_{out}, l_s) function ensemble $\mathcal{F} = \{F_n\}_{n \in \mathsf{N}}$ $F_n = \{f_s : \{0,1\}^{l_{in}(n)} \rightarrow \{0,1\}^{l_{out}(n)}\}_{s \in \{0,1\}^{l_s(n)}}$, $Eval_{\mathcal{F}}$ ( ) $s \in \{0,1\}^{l_s(n)}$ $x \in \{0,1\}^{l_{in}(n)}$, $Eval_{\mathcal{F}}(s, x) = f_s(x)$ s seed f_s

Definition 2. (l_{in}, l_{out}, l_s) $\mathcal{F}$ pseudorandom, D ( ), $p(\cdot)$, $n \in \mathbb{N}$, $z \in \{0,1\}^*$, $|\Pr[s \leftarrow \{0,1\}^{l_s(n)}; b \leftarrow D^{f_s}(1^n, z) : b = 1] - \Pr[u \leftarrow U_{l_{in}(n), l_{out}(n)}; b \leftarrow D^u(1^n, z) : b = 1] < \frac{1}{p(n)}$

2.1 Private-Key Encryption

We consider block-ciphers which can operate on plaintexts of a specific length. A private-key encryption scheme consists of three algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation algorithm $\mathcal{K}$ is a randomized algorithm that takes as input the security parameter 1^n and returns a common private key K. The encryption algorithm $\mathcal{E}$ is a randomized algorithm that takes a key K and a plaintext $M \in \{0,1\}^{poly(n)}$ to return a ciphertext C, where $poly(\cdot)$ is a fixed polynomial representing the length of plaintexts. The encryption process is denoted by $C \leftarrow \mathcal{E}(K, M)$. When we need to make explicit the random coins R used by $\mathcal{E}$, we write $C = \mathcal{E}(K, M, R)$. The decryption algorithm $\mathcal{D}$ is a deterministic algorithm that takes a key K and a ciphertext C to return the plaintext M, and the decryption process is denoted by $M = \mathcal{D}(K, C)$. It is required that, for every n and every plaintext $M \in \{0,1\}^{poly(n)}$, $\Pr[K \leftarrow \mathcal{K}(1^n); C \leftarrow \mathcal{E}(K, M) : \mathcal{D}(K, C) = M] = 1$.

When we consider a fixed key K, we denote by $\mathcal{E}_K(\cdot, \cdot)$ an encryption program with the key K embedded. We can view such a program as a boolean circuit.

We review the standard private-key encryption scheme based on (l_{in}, l_{out}, l_s)-pseudorandom function ensembles [7, Construction 5.3.9].

- $\mathcal{K}(1^n)$ uniformly generates a string K of length $l_s(n)$.
- $\mathcal{E}(K, M, R) = (M \oplus f_K(R), R)$, where R and M are of length $l_{in}(n)$ and $l_{out}(n)$, respectively. We denote by C_1 the first part of ciphertexts $(M \oplus f_K(R))$ and by C_2 the second part (R).
- $\mathcal{D}(K, (C_1, C_2)) = C_1 \oplus f_K(C_2)$

We don't formally define the security definitions of encryption schemes, but we informally recall the security against chosen-plaintext attacks. A private-key encryption scheme is secure against chosen-plaintext attacks if no adversary with oracle access to an encryption program[3] (with a randomly generated key embedded) can pick two plaintexts (of equal length) such that it can distinguish between ciphertexts of the two plaintexts. The above standard scheme is secure against chosen-plaintext attacks [7, Section 5.4.3].

As discussed in Section 1.2, any private-key encryption scheme can be transformed into a (candidate) public-key encryption scheme by publishing as the public key an obfuscation of an encryption program with a key K, i.e., $\mathcal{O}(\mathcal{E}_K)$. It is easy to see that if a private-key encryption scheme is insecure against chosen-plaintext attacks, then the resulting public-key encryption scheme is insecure as well (against chosen-plaintext attacks). Therefore, for the transformation to work, original private-key encryption schemes must be secure against chosen-plaintext attacks.

2.2 Obfuscators

We recall the definitions of obfuscators in [1]. Barak et al. deal with two kinds of obfuscators; one is for Turing machines (TMs) and the other is for boolean circuits. Here we focus only on circuit obfuscators due to a subtlety we have in TM obfuscators (although our argument to be followed basically applies to TM obfuscators as well). Note that TMs can take input of arbitrary length. Therefore, given a description of a TM that computes an encryption algorithm (with a private key embedded), it is easy to modify it so that, if it takes an irregular input (such as an empty string and strings shorter or larger than expected), it outputs the private key. Clearly, no matter how we obfuscate the modified TM description, the resulting public-key encryption scheme is insecure. Due to this subtlety, we focus only on circuit obfuscators.

We focus only on syntactic (functionality) requirements because our argument won't depend on any specific security requirement, i.e., how to define "unintelligible" (as in the negative result of [1]). As discussed in Section 1, we consider two kinds of functionality requirements: (1) circuit obfuscators and (2) sampling obfuscators.

Definition 3 (Circuit Obfuscator). [illegible] $\mathcal{O}$ [illegible] circuit obfuscator [illegible] C, $\mathcal{O}(C)$ [illegible] C [illegible] 1 ([illegible] $\mathcal{O}$), [illegible] $\Pr[C' \leftarrow \mathcal{O}(C) : \forall x, C(x) = C'(x)] = 1$

Take for example the above standard encryption scheme. Let $\mathcal{E}_K(M, R)$ be an encryption program with a key K embedded. We view it as a boolean circuit. Let $\mathcal{E}'_K(M, R)$ be a program obfuscated by a circuit obfuscator. It requires that $\mathcal{E}_K(M, R) = \mathcal{E}'_K(M, R)$ for every pair (M, R).

[3] In the definition of chosen-plaintext attacks, adversaries are NOT allowed to choose the random input when they use oracle access to encryption programs. That is, the oracle internally flips its coins. However, in this paper, when we talk about oracle access to obfuscated encryption programs, we allow adversaries to set the random input. See Remark 1.

[1, Section 4.2] considers obfuscators which has a relaxed functionality requirement in the sense that we allow the obfuscated circuit to only the original circuit. Since the relaxed definition would cause decryption errors in resulting public-key encryption schemes, we don't deal with it in this paper.

Next, we define sampling obfuscators. Here, we distinguish between the regular input and the random input to circuits. Let $C(x, r)$ be a circuit which takes the regular input x and the random input r. Given a regular input string x, we can view $C(x, \cdot)$ as a sampling algorithm for the distribution obtained by evaluating $C(x, r)$ on random coins r.

Definition 4 (Sampling Obfuscator). $\mathcal{O}$ sampling obfuscator C, $\mathcal{O}(C)$ 1 ($\mathcal{O}$) x, $C(x,r)$ $C'(x,r)$ C' $\mathcal{O}(C)$

Again, take for example the above standard encryption scheme. Let $\mathcal{E}_K(M, R)$ be an encryption program with a key K embedded. This time, we view M as the regular input and R as the random input, respectively. Let $\mathcal{E}'_K(M, R)$ be an obfuscated program of it. It requires that the two distributions obtained by evaluating $\mathcal{E}_K(M, R)$ and $\mathcal{E}'_K(M, R)$ on independent random coins R are the same.

We can relax the definition by allowing the distribution by the obfuscated circuit is statistically close to the distribution by the original circuit. However, we don't deal with it because it would cause decryption errors in resulting public-key encryption schemes.

In order to prove our result regarding sampling obfuscators, we need to require that the obfuscation preserves an input-output dependency of circuits in the following sense: We partition the output of a circuit into two: (1) the part whose value depends on the random input, but not on the regular input and (2) the other part.

Definition 5 (Sampling Obfuscator that Preserves Input-Output Dependency). C, $RO(C)$ $\mathcal{O}$ preserves input-output dependency C, $RO(\mathcal{O}(C))$ 1 ($\mathcal{O}$)

Again, take for example the above standard encryption scheme. Let $\mathcal{E}_K(M, R) = (M \oplus f_K(R), R)$ be an encryption program with a key K embedded, where M is the regular input and R is the random input. Let $\mathcal{E}'_K(M, R) = (M \oplus f_K(R'), R')$ be an obfuscated program of it, where the output R' may not be equal to the input R. If the obfuscator preserves input-output dependency, then R' should be computed only from R.

3 Our Results

We say that a probabilistic private-key encryption scheme is [illegible] by circuit (resp., sampling) obfuscators if, for every circuit (resp., sampling) obfuscator, the resulting public-key encryption scheme is [illegible] in the sense that, given any public key, one can efficiently decrypt every ciphertext (encrypted under the public key). In this section, we construct two untransformable [illegible] private-key encryption schemes. One is untransformable by circuit obfuscators (Section 3.1) and the other is by sampling obfuscators that preserve input-output dependency (Section 3.2). When we show that the resulting public-key encryption schemes are totally insecure, we use obfuscated encryption programs (i.e., public keys) [illegible]. Therefore, we can say that they are untransformable even if they are obfuscatable under a definition of the virtual black-box property (Section 3.3).

Our results assume the existence of pseudorandom function ensembles, i.e., the existence of one-way functions. Also, recall that we allow adversaries to set the random coins when they execute a given public key (an obfuscated encryption program).

3.1 Private-Key Encryption Untransformable by Circuit Obfuscators

Theorem 1. [illegible]

[illegible]

[illegible]

Proof: We consider the standard private-key encryption scheme based on pseudorandom functions [7, Construction 5.3.9]. $\mathcal{E}_K(M, R) = (M \oplus f_K(R), R)$. We denote by C_1 the first part of ciphertexts $(M \oplus f_K(R))$ and by C_2 the second part (R). It is secure against chosen-plaintext attacks [7, Section 5.4.3].

We note that the ciphertext (C_1, C_2) can be decrypted given oracle access to any encryption program, i.e., $\mathcal{E}_K(C_1, C_2)$ outputs (M, C_2). Therefore, for every key K and every circuit obfuscator $\mathcal{O}$, the obfuscated program $\mathcal{E}'_K \leftarrow \mathcal{O}(\mathcal{E}_K)$ can be used as a black-box to decrypt every ciphertext because $\mathcal{E}_K(M, R) = \mathcal{E}'_K(M, R)$. □

[illegible] We note that the argument used to construct the [illegible] unobfuscatable scheme in [1, Section 4.3] can be easily extended to construct a [illegible] unobfuscatable scheme as long as we consider circuit obfuscators. That is, their argument implies Theorem 1. However, as mentioned in Section 1.3, it is unclear how to extend the argument to obtain a [illegible] scheme that is unobfuscatable even by [illegible].

[illegible] The argument might remind readers of the notion of weak keys [4]. In the case of deterministic encryption schemes, we mean by a weak key a key under which encryption and decryption are the same function. In the case of probabilistic encryption schemes, we mean a key under which encryption algorithm can be used for decryption as in our argument. Our result can be generalized to a class of secure private-key encryption schemes such that the fraction of weak keys is not negligible (although we need to extend our definitions so that we can take into account the distribution of keys).

3.2 Private-Key Encryption Untransformable by Sampling Obfuscators That Preserve Input-Output Dependency

The argument in the proof of Theorem 1 does not go through when we use [illegible] for the transformation. In this section, we will construct a probabilistic private-key encryption scheme that is untransformable by sampling obfuscators that preserve input-output dependency.

Theorem 2. [illegible]

[illegible]

Proof: We modify the standard scheme using an additional randomness R_2 as follows:

$$\mathcal{E}(K, M, (R_1, R_2)) = (M \oplus f_K(R_1), R_1, f_K(R_2 \oplus M), R_2)$$

Note that the first half of the ciphertext is the same as the ciphertext of the standard scheme, and the second half is appended. The key generator is the same. We use the same decryption algorithm although we don't use the second half of the ciphertext for decryption.

The modified scheme is secure against chosen-plaintext attacks by the same argument in [7, Proposition 5.4.12].

We will use the second half of the ciphertext to show that oracle access to any obfuscated program $\mathcal{E}'_K(M, (R_1, R_2))$ that samples the same distribution as $\mathcal{E}_K(M, (R_1, R_2))$ can be used to compute the pseudorandom function f_K as long as the obfuscator preserves input-output dependency. Given an input string x, we can use such an oracle access to compute $f_K(x)$ as follows:

Step 0. Let M be any plaintext and (R_1, R_2) be any set of random coins.

Step 1. Invoke $\mathcal{E}'_K$ on $(M, (R_1, R_2))$ to obtain $(f_K(R'_2 \oplus M), R'_2)$, where R_2 may not be equal to R'_2.

Step 2. Invoke $\mathcal{E}'_K$ on $(x \oplus R'_2, (R_1, R_2))$ to obtain $(f_K(x), R'_2)$, where M is replaced by $x \oplus R'_2$ and the same random input is used. Note that, because the input-output dependency of $\mathcal{E}_K$ is preserved, R'_2 at Step 1 and 2 are the same.

Step 3. Output $f_K(x)$.

It is easy to see that we can efficiently decrypt every ciphertext using the ability to compute f_K. □

3.3 Discussion

Our argument in the two theorems uses only oracle access to obfuscated programs while the argument in [1, Section 4.3] needs to use an obfuscated program itself. Therefore, the transformation from the two private-key encryption schemes does not work at all [illegible]. In this sense, our result would be more negative than the result of [1]. On the other hand, the positive result of [14] says that a probabilistic private-key encryption scheme secure against chosen-plaintext attacks can be transformed a probabilistic public-key encryption scheme secure against chosen-plaintext attacks [illegible]. Readers may wonder if our results contradict the positive result. However, it is not the case because the virtual black-box property defined in [14] is different from the one we discuss in this paper. Recall that the virtual black-box property requires that whatever one can efficiently compute given an obfuscated program can be computed given oracle access to the functionality. What do we mean by "given oracle access to a functionality" when the functionality is probabilistic? There are two distinct meanings[4]. In this paper, it means that we can choose both the regular and random inputs and tell the functionality to compute the output from the two inputs. On the other hand, in [14], it means that we can choose only the regular input and that the functionality internally flips its coins. As a result, the virtual black-box property defined in [14] would be stronger than the one we discuss in this paper. Therefore, even if we can prove Theorem 2 without assuming that sampling obfuscators preserve input-output dependency (still by using only oracle access to obfuscated programs), it won't contradict the positive result of [14].

4 Concluding Remarks

We have shown that there is a [illegible] private-key encryption scheme that is untransformable by sampling obfuscators that preserve input-output dependency. Interestingly, the scheme is untransformable even if it is obfuscatable under a definition of the virtual black-box property.

Our result implies that, general-purpose obfuscators for the transformation from private-key encryption into public-key encryption, if they exist, must be a sampling obfuscator which does not preserve input-output dependency. In other words, it would be an interesting open question whether we can construct a [illegible] private-key encryption scheme that is untransformable even by sampling obfuscators that do NOT preserve input-output dependency.

Acknowledgments

We would like to thank anonymous reviewers of IWSEC'2007 for helpful comments.

[4] The same distinction is discussed in the context of black-box simulation zero-knowledge in [9].

References

1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (Im)possibility of Obfuscating Programs ECCC, Report No. 57, 2001. (In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001))
2. Canetti, R.: Towards Realizing Random Oracles: Hash Functions that Hide All Partial Information. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997)
3. Canetti, R., Micciancio, D., Reingold, O.: Perfectly One-way Probabilistic Hash Functions. In: Proceedings of 30st STOC (1998)
4. Davies, D.W.: Some Regular Properties of the DES. In: McCurley, K.S., Ziegler, C.D. (eds.) CRYPTO 1982. LNCS, vol. 1440, Springer, Heidelberg (1982)
5. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Trans. Inform. Theory 22(6), 644–654 (1976)
6. Dodis, Y., Smith, A.: Correcting Errors without Leaking Partial Information. In: Proceedings of 37th STOC (2005)
7. Goldreich, O.: Foundations of Cryptography: Volume II Basic Applications. Cambridge University Press, Cambridge (2004)
8. Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. Journal of the ACM 33(4), 792–807 (1986)
9. Goldreich, O., Oren, Y.: Definitions and Properties of Zero-Knowledge Proof Systems. Journal of Cryptology 7(1), 1–32 (1994)
10. Goldwasser, S., Kalai, Y.T.: On the Impossibility of Obfuscation with Auxiliary Input. In: Proceedings of FOCS 2005, pp. 553–562 (2005)
11. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. System Sci. 28, 270–299 (1984)
12. Goldwasser, S., Rothblum, G.N.: On Best-Possible Obfuscation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (2007)
13. Hada, S.: Zero-Knowledge and Code Obfusacation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 443–457. Springer, Heidelberg (2000), http://www.springer.com/east/home/generic/search/results?SGWID=5-40109-22-2128743-0
14. Hofheinz, D., Malone-Lee, J., Stam, M.: Obfuscation for Cryptographic Purposes. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (2007)
15. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely Obfuscating Re-Encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (2007)
16. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proceedings of 21st STOC (1989)
17. Lynn, B., Prabhakaran, M., Sahai, A.: Positive Results and Techniques for Obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, Springer, Heidelberg (2004)
18. Wee, H.: On Obfuscating Point Functions. In: Proceedings of STOC 2005, pp. 523–532 (2005)

Design Issues of an Isolated Sandbox Used to Analyze Malwares

Shinsuke Miwa[1,3], Toshiyuki Miyachi[2,3], Masashi Eto[1],
Masashi Yoshizumi[2,4], and Yoichi Shinoda[1,2,3]

[1] National Institute of Information and Communications Technology,
Information Security Research Center, 4-2-1 Nukui-Kitamachi, Koganei, Tokyo
184-8795 Japan
danna@nict.go.jp
http://www2.nict.go.jp/y/y201/src-web/index-e.html
[2] National Institute of Information and Communications Technology,
Hokuriku Research Center, 2-12, Asahidai, Nomi-city, Ishikawa 923-1211 Japan
[3] Japan Advanced Institute of Science and Technology, Internet Research Center,
1-1 Asahidai, Nomi, Ishikawa 923-1292 Japan
[4] Japan Advanced Institute of Science and Technology, School of Information Science,
1-1 Asahidai, Nomi, Ishikawa 923-1292 Japan

Abstract. Recent viruses, worms, and bots, called malwares, often have anti-analysis functions such as mechanisms that confirm connectivity to certain Internet hosts and detect virtualized environments. We discuss how malwares can be kept alive in an analyzing environment by disabling their anti-analyzing mechanisms. To avoid any impacts to/from the Internet, we conclude that analyzing environments should be disconnected from the Internet but must be able to make malwares believe that they are connected to the real Internet. We also conclude that, for executing environments to analyze anti-virtualization malwares, they should not be virtualized but must be as easily reconstructable as a virtualized environment. To reconcile these cross-purposes, we propose an approach that consists of a mimetic Internet and a malware incubator with swappable actual nodes. We implemented a prototype system and conducted an experiment to test the adequacy of our approach.

1 Introduction

Malwares[1], such as viruses, worms, and bots, are daily becoming more sophisticated. To confront the malwares threat, it is necessary to observe their behavior, analyze their mechanisms, and identify issues. Isolated sandboxes are good analyzing environments for carrying out such observations because they have tolerance to attacks and infections from the outside.

Isolated sandboxes are now easy to build because of improvements in technologies such as OS and hardware virtualizations. Since malwares often damage analyzing environments, the environments must frequently be rebuilt. Because of this, virtualization technologies have been widely used because analyzing environments introduced to a virtualization technology are easy to rebuild. We had

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 13–27, 2007.
© Springer-Verlag Berlin Heidelberg 2007

developed a isolated sandbox based on a VMM (Virtual Machine Monitor) that called “VM Nebula.”[2]

Unfortunately, some recent malwares have anti-analysis mechanisms for examining the environment in which they are being executed. Therefore, such malwares cannot be efficiently analyzed in isolated sandboxes and virtualized environments because they are capable of detecting analyzing environments. Our developed “VM Nebula” could not clearly avoid these issues.

We discuss how malwares that have anti-analysis mechanisms can be safely live observed, analyzed, and identified in analyzing environments. On the basis of the discussion, we propose two approaches. The first is the mimetic Internet, which cannot be recognized as an isolated environment by these malwares. The second is the malware incubator that is based on swappable actual (not virtualized) nodes for easy reconstruction. We also implemented a prototype system that incorporates our approaches and used it to conduct a proof-of-concept experiment to test its efficiency and clarify issues in using it.

In this paper, we will use the following terminology: an experimental environment for analyzing malwares is called an analyzing environment, a malware on an executable format is called an executable instance, and an environment for executing a malware is called an executing environment. A mechanism designed to avoid any impacts to/from the outside, i.e., the Internet, from/to the analysis in an isolated analyzing environment is called an isolated sandbox. Finally, a virtualized computing environment with OS virtualization technologies or hardware virtualization technologies is called a virtualized environment.

2 Anti-analysis on Malwares

As mentioned above, mechanisms to disrupt analysis have been introduced on the latest malwares. In this section, we describe these anti-analysis mechanisms.

Anti-analysis mechanisms can be classified into two types:

- Obfuscate executable instances
- Restrict executing environments

We describe these briefly below.

2.1 Obfuscating Executable Instances

The purpose of obfuscating executable instances is to disrupt static code analysis with offline reversing tools such as disassemblers and debuggers. The two major techniques for disrupting static analysis[3] are:

- Program code obfuscation
- Binary code transformation

These techniques have often been combined to enhance anti-analysis capability. In this paper, we will not discuss elaborate on them because the static analysis is not our target.

2.2 Restricting Executing Environments

The purpose of restricting executing environments is to disturb live code analysis by analyzing environments such as isolated sandboxes and virtualized environments. The two major techniques are:

1. Detecting executing environment
2. Controlling execution and hiding instance

These malware techniques are usually used sequentially, with the first being used to determine whether or not the second must be implemented.

Detecting an Executing Environment. When a malware detects an executing environment, it decides whether to execute itself in an analyzing environment. The malware checks whether it is stepping on debuggers or executing itself in isolated sandboxes or in virtualized environments. Then, it carries out a controlling execution of itself and a hiding instance of itself based on the result of its decision to prevent live code analysis.

Certain countermeasures for debugger analysis, such as inspecting abnormal running time, which can detect step execution on debuggers, and overwriting the interrupt vector table for watch points, which can cancel watch point debugging, are well-known[3]. These countermeasures have been used to protect software. Malwares, however, have been introducing countermeasures to prevent analyzing of themselves.

Two common countermeasures used by malwares against isolated sandboxes are confirming the executing environment's IP address and checking reachability to the Internet from the environment. Some malwares use the former countermeasure to check whether or not their own IP addresses on the executing environment are private IP addresses because isolated sandboxes often use IP addresses on private IP address spaces. Malwares use the latter countermeasure to check whether specific targeted hosts and services are reachable because isolated sandboxes often are disconnected from the Internet. These countermeasures are widely used because they are easy to implement. They are also essential functions for some malwares that must connect to the Internet to download their own main bodies while it is running and to join the commandment network.

Virtualized environments counter malware mechanisms by detecting whether or not malware executes itself in a virtualized environment with virtualization technologies. One countermeasure is to use special processor codes (*cf.* section 3) that cause results that differ for malwares that are and are not in a virtualized environment with processor virtualization technologies. Another countermeasure is to match BIOS banners and device names that are specific to each virtualized environment. In recent years, particular malwares have been introducing these countermeasures because virtualized environments have been widely used as analyzing environments.

Controlling Execution and Hiding Instance. A malware controls its execution and hides its instance by disturbing the live code analysis when it detects a poorly executed environment, such as an analyzing environment that may be able to analyze it, or an environment too insignificant for its execution.

Common techniques for malwares to control their execution are changing their behavior to non-native behavior by, for example, impersonating well-known attacking behavior, and halting their execution. Many malwares halt their execution when they determine that they are in an insignificant environment.

Some malwares hide their executable instances to disrupt collection of the unpacked instance, which could be used to analyze them. Common techniques for hiding an executable instance are eliminating the instance of itself and implanting itself or a part of itself. These are set up to restrictively execute to the disk area for system or system files based on system events. Trojan horses, a particular kind of virus, have long been using these techniques, and the latest malwares now also use them.

3 Related Work

Honeypot technologies such as the Honeynet project[4] are major technologies for mimicking sites and hosts on the Internet. Honeypots mimic an analyzing environment as generic hosts and local sites to intruders. The Honeynet project has been developing mimicking technologies with virtualization technologies and mimicking techniques for large-scale sites, which have been using them to analyze actual incidents. Our approach is related to Honeypot technologies, but has some differences. Our current target is automatic intrusions by malwares, and our interest is mimicking parts of the Internet.

Virtualization technologies[5] build a virtualized computer in an actual computer. Therefore, they should provide the following functions:

– A single computer should perform as well as multiple computers.
– A virtualized environment, which is generally called a "guest environment," should be easily managed by a virtualization technology's executing environment, which is generally called a "host environment."
– The host environment should be easily concealed from the guest environment.

Because of these functions, virtualization technologies have been widely used on high-availability server technologies and testbeds for testing and experimenting with systems.

In a recent year, support mechanisms for virtualization technologies have been introduced on some processors for personal computers. In response to this, malware techniques that conceal themselves using processor virtualization technologies[6] have been developed. These techniques cannot always be detected by a host OS because host OSs sometimes execute in ultra thin host environments[7] prepared by the malware. To counter these techniques, techniques for detecting a virtualized environment using specific processor instructions have been proposed[8]. Different results are returned by these counter techniques when they are running in a virtualized environment. Techniques for avoiding[9] detection have also been discussed. In other words, a cat-and-mouse game is developing into a vicious spiral.

Strider HoneyMonkeys[10] is capable of exploit detection with virtualization technologies. To detect exploitation, it drives the Internet Explorer browser and some programs in a way that mimics a human user's operation, and analyzes behaviors of the drived programs. Many web sites that exploit browser vulnerabilities could be found with it. Although the HoneyMonkeys seems to be efficient to detect exploitation, yet it seems to have issues of anti-virtualization malwares as mentioned the above.

Potemkin Virtual Honyefarm[11] is a honeyfarm with XEN[12] virtual machine monitor which modified to support their virtual honeyfarm architecture. Potemkin goal is high-fidelity honeypots over a large number of IP address, and prototype Potemkin implementation achived high-fidelity host emulation, based on the paravirtualization of modified XEN, for hundreds of thousands of IP address while using only tens of physical servers. Although the Potemkin seems to provide real Internet connectivity and high-fidelity emulated hosts to execute specimens, yet it seems to have issues of anti-virtualization malwares as same as the HoneyMonkeys.

4 Design Issues

To account for the anti-analysis mechanisms in the malwares mentioned above, we focused on techniques of detecting executing environments. In this section, we will discuss how we can fool the detection.

This paper focuses on developing analyzing environments. Methodologies of static and live code analysis, such as using debuggers, are outside the focus of this paper.

4.1 Requirements

Live malware codes might attempt to infect and attack other hosts and sites based on the malware's purposes. Therefore, when there are live codes in a sandbox, the sandbox helps the infections and attacks. The Internet enables attacks. If an analyzing environment is directly connected to the Internet, it will suffer from impacts from the Internet. Therefore, environments for analyzing live malware codes must be isolated from the Internet to avoid impact to/from the Internet.

Since malwares often damage analyzing environments, to observe malware behavior, the analyzing environment must be rebuilt to recover from damages caused by the malware. Therefore, analyzing environments must be easy to rebuild. They must also be constructed so that they are able to fool the detecting mechanisms of malwares:

1. Check whether or not specific targeted hosts and services are reachable
2. Detect virtualized environments

There is no simple approach to the reachability check (1) technique because providing reachability from the isolated sandbox causes a troublesome conflict between isolating from and connection to the Internet. Two common approaches are capable of reconciling the conflict. The first, called the online approach, provides reachability to specific hosts and services on demand. In this approach, curious and intelligent traffic filters must be provided because attacking and infecting traffic from malware must not be able to pass through to the Internet. The second, called the offline approach, provides counterfeit reachability. In this approach, counterfeit hosts and services must be prepared based on the malware's targets.

There are also two fundamental approaches to the virtualized environment detection (2) technique. The first is making virtualized environments seem not virtualized. The second is providing the same functions as virtualized environments provide in non-virtualized environments. In the second approach, the following advantages must be kept in an executing environment based on actual nodes instead of virtualized environments. Virtualized analyzing environments have the advantages of:

- making it easy to construct an isolated sandbox because a virtualized environment usually has a closed executing environment,
- making it easy to rebuild an environment using snapshotting itself, and rolling back previous states which are provided in many implementations of the virtualization technology.

Based on these issues, we discuss below how to make an isolated sandbox appear to be connected to the real Internet and how to retain the advantages of virtualized environments in an executing environment based on actual nodes.

4.2 End-Host View of the Internet

A function that enables an analyzing environment to pass a malware's reachability check even though it is executed in an isolated sandbox is called "mimetic Internet." To construct a mimetic Internet, what is important is whether or not malwares mistake the mimetic Internet for the real one. This may amount to what could be observed in the real Internet from an end-host and how our mimetic Internet mimics the real Internet to the end-host.

An end-host should be able to observe the behavior of:

1. neighboring hosts on the same network segment,
2. local services and server hosts on the same site,
3. global services and server hosts on the Internet,
4. and opened hosts on the Internet.

It must also be able to observe environmental parameters on the network routes from the communications with them. Environmental parameters such as

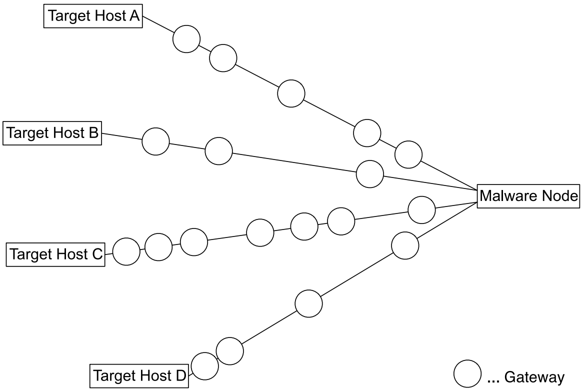


Fig. 1. End-host view of Internet

bandwidth and round trip time (RTT) of an observed target host, a count of transit gateways, and RTTs to each gateway can be observed by the end-host. However, it is difficult, based only on these environmental parameters, for the end-host to get an overview of the Internet and where it is because the parameters are independent for each target host.

We can therefore conclude that the end-host view of the Internet could be modeled as a collection of behavior of target hosts involving some services, a count of gateways, and link qualities such as bandwidths and RTTs, for each target host and gateway. Figure 1 shows this model. In the sections below, we use the term target host to mean the targeted host of malwares checking reachability.

4.3 Actual Nodes Instead of Virtualized Environment

The executing environment based on actual nodes would not ordinarily be able to provide an isolated executing environment that could ordinarily be provided in virtualized environments. Also, when actual nodes are used, it usually takes more time and work for the nodes to be configured and for the executing environment to be built based on the nodes.

In the sections below, the term “actual node” is used to mean an actual computer that is not virtualized.

Isolated Sandbox Based on Actual Nodes. An isolated sandbox based on actual nodes can easily be built to function as a strictly disconnected analyzing environment. However, in this disconnected environment, management of the environment through actions such as introducing and executing specimens and observations such as capturing packet dumps and collecting related logs must be carried out without network connections to the environment. Thus, in constructing the isolated sandbox based on actual nodes, the manner in which the analyzing environment is permitted to control and observe the isolated sandbox is important.

Rebuilding Environment Based on Actual Nodes. In live code analysis of malwares, an executing environment must frequently be rebuilt to recover from damage caused by malwares. This is time consuming and difficult. Though the amount of work could be reduced by using support software, the amount of time required cannot easily be reduced.

5 Approaches and Prototyping

In light of the design issues mentioned above, we designed a prototype system to implement our approaches. In this section, we describe our approaches and the design of the prototype system.

5.1 Our Approaches

In response to the reachability check technique mentioned in section 4.1, we use the offline approach, which fools reachability checks by using a function that we call the mimetic Internet. We chose this approach because, with the online approach, it is difficult to eliminate the risk of violating isolation because traffic filters not only pass through traffic for the reachability check but also attack and infect traffic.

For the virtualized environment detection technique, we use a non-virtualized environment that is capable of providing the same functionality as a virtualized environment. This is because techniques for concealing virtualizations probably interact with techniques for detecting concealed virtualizations, causing a vicious circle (*cf.* section 3). We attempt to avoid the vicious circle.

5.2 Functions

The function called "mimetic Internet" must mimetically provide targeted hosts, services, and network environments. Some functions must be provided because the advantages of virtualized environments must be kept in an executing environment based on actual nodes instead of in virtualized environments. These functions are called the "malware incubator."

Figure 2 shows a functional image of our proposed environment. We discuss these functions in detail below.

5.3 Mimetic Internet

According to the model of the end-host view of the Internet (*cf.* section 4.2), if we could emulate the behavior, number of gateways, and link qualities of each target host and gateway, malwares would misidentify as themselves on the real Internet. Therefore, our proposed mimetic Internet will be composed of:

- Mimetic target hosts
- Emulated routes

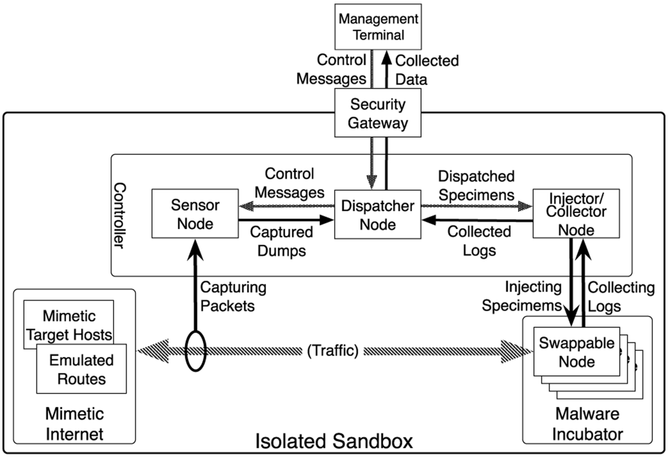


Fig. 2. Functional image of proposed environment

Mimicking Target Hosts. If a malware has information about all the hosts on the Internet and can check them all, the same number of hosts must be mimicked on a mimetic Internet. The malware, however, cannot inspect all the hosts on the Internet because there is far too much information involved. Therefore, malwares should be able to choose target hosts from the following classes and combinations of them:

1. hosts that are easy to inspect,
2. principal or fundamental hosts that are always present, and
3. hosts specific to the malware.

Accordingly, the proposed system uses mimetic target hosts from each class. For the easy-to-inspect (1) class, neighboring hosts on the same network segment and server hosts for major local services such as SMTP, POP, Web, and DNS, which are provided on generic sites, were prepared. For the principal or fundamental (2) class, global DNS services such as root DNSs, famous search engines *Yahoo!* (`http://www.yahoo.com/`) and *Google* (`http://www.google.com/`), and `time.windows.com`, which is a default NTP server for Microsoft Windows XP hosts, were prepared. A dynamic configuration of specific target hosts based on the results of a pre-experiment that executes the malware and collects communication history were discussed as examples of the class (3) specific to the malware.

Emulating Routes. On the real Internet, an observed route would perhaps be changed for each observation because the route would be chosen by routing mechanisms from many possible routes. However, the observed route usually has only a single gateway count because there usually is no other route to the target host, and the route should be stably chosen.

The proposed method emulates a route to a target host using the route emulator and the link quality emulator. The route emulator emulates as each target host transits a fixed count of gateways, and the link quality emulator emulates bandwidths and delays to each gateway using a network emulation software.

5.4 Malware Incubator

The malware incubator cannot ordinarily provide an isolated executing environment, and an actual node usually takes more time and work to rebuild. To solve these problems, our method was designed as discussed below.

Isolated Sandbox Based on Actual Nodes. Our proposed method provides the isolated sandbox with separated VLANs, a security gateway, and sensor nodes. The analyzing environment is separated on specified VLANs, which are disconnected from other environments, and the security gateway, which is built up between the isolated sandbox and a management terminal, provides communication channels for controlling and observing the isolated sandbox. Sensor nodes provide captured packet dumps with a packet capture software and collect related logs via `FTP` or `SCP`.

Rebuilding an Environment Based on Actual Nodes. In our method, the environment would be built up on large-scale practical estimation testbeds, such as the StarBED[13], to solve these problems. The testbed is composed of a large number of the same kind of actual nodes and provides easily changing network topologies and strictly controls experimental steps with its supporting software. This supporting software should reduce the amount of work required.

However, the time required cannot easily be reduced. Therefore, an enormous number of the same kind of actual nodes must be prepared on the large-scale testbed. The proposed method will use a node swapping technique. In this technique, the same executing environments would be previously installed on many actual nodes. Then an actual node can be swapped to other nodes installed in the same executing environment when the executing environment sustains damage. After swapping, the damaged executing environment can be rebuilt in the background.

Figure 3 shows an overview of the node swapping method. First, executing environment A for a malware is installed on actual node 1. Then, the same environment would also be installed on actual nodes 2 and 3 as spares. After executing the malware, damage would be done to environment A on node 1. Then, executing environment A could be swapped from node 1 to node 2, and the malware would be continuously analyzed. After being swapped, environment A could be rebuilt on node 1 in the background of the analysis.

In this method, the time required to rebuild, which naturally involves stopping the analysis, could be reduced because executing environments would be swapped to already built spare nodes and be rebuilt in the background.

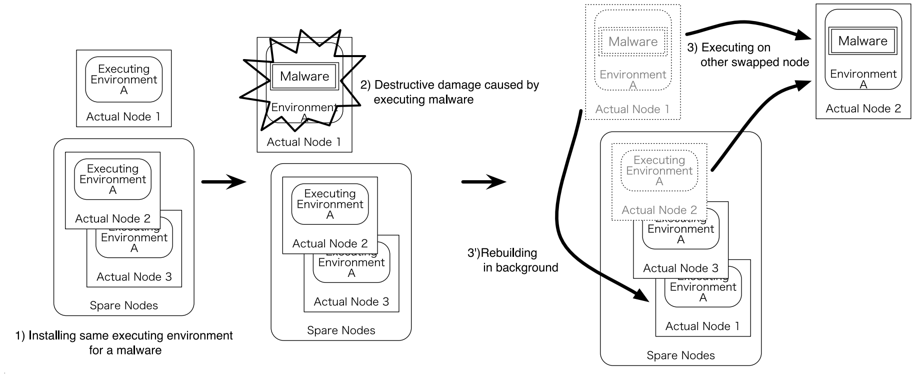


Fig. 3. Overview of Node swapping

6 Evaluation

We conducted a proof-of-concept experiment to test the efficiency and clarify issues of our approaches. In this section, we describe the experiment and its results.

6.1 Experiment

The purposes of the experiment were to determine what kinds of malwares the mimetic Internet could fool and how much the time between each re-experiment could be reduced by the swappable actual nodes method.

To determine these things, a conceptual prototype of our analyzing environment was constructed on the StarBED. Then, some unknown specimens, which had been collected in our *Nepenthes*[14] environment were filtered out by *ClamAV*[15]. We observed the behavior of their live codes in the environment with `tcpdump`, `snort`[16], and system monitoring tools for Microsoft Windows [17](FileMon, RegMon, TCPview, and Process Explorer).

Figure 4 shows an overview of the environment. The environment included:

- Five actual nodes, which were installed in non-patched Microsoft Windows XP professional for the executing environment that included four spare nodes
- A sensor node using `tcpdump` and `snort`, which was in bridge mode, for capturing packets, and a sensor node for injecting specimens
- A mimetic local site with four actual nodes that were installed in fully patched Windows, SMTP/POP/Web/DNS local servers, and a firewall router
- Four simple link quality emulators with `qdisc` (netem)[18] to emulate routes to mimetic global servers and sites
- Mimetic *time.windows.com*, which is a default NTP server for Windows XP, as an NTP server on the mimetic global server
- Mimetic *Google* and *Yahoo!* as web servers on the mimetic major sites

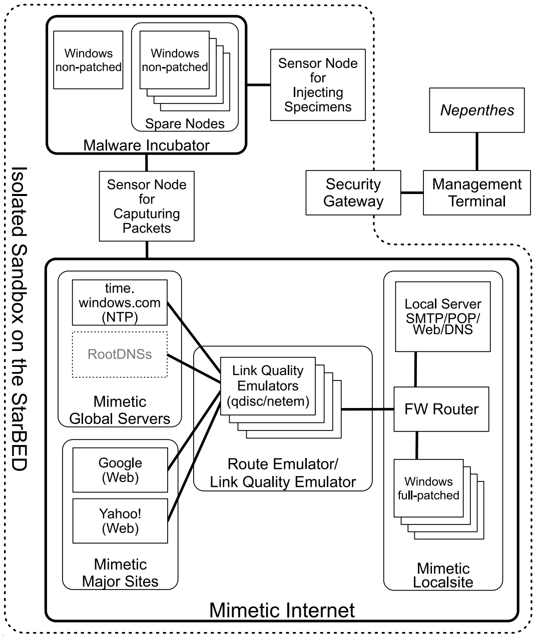


Fig. 4. Overview of Experimental Environment

– An isolated sandbox, which is composed of all of the above
– The security gateway, which is a simple firewall router

And note that there are no support mechanism for specific mimetic hosts for each malware because no mechanism for dynamic configuration of specific targets was implemented.

The isolated sandbox must be disconnected from the Internet. In this experiment, it was isolated using separated VLANs and physically disconnected wires. We verified its separateness with a penetration test using `nmap`[19].

Table 1 shows classification of collected specimens. We had been collecting specimens, which included 23 unknown specimens, on *Nepenthes* from December 27, 2006 to February 16, 2007. We analyzed 14 specimens, all of which are valid Windows executables.

We should noticed that this experiment could not verify reducing works because we manually constructed the isolated sandbox, injecting specimens and collecting data for the analysis of this experiment.

6.2 Results

Results are shown in Table 2. We observed that six specimens, fooled by our mimetic global sites and global servers, accessed our mimetic Internet and that their behavior changed based on whether or not they were fooled. In the observations, the expected behavior, accessing *Google* or *Yahoo!* via HTTP and

Table 1. Classification of Specimens

Classifications	Counts
Collected specimens on our *Nepenthes*	1069
Unified specimens by MD5	163
Unknown specimens (filtered out by *ClamAV*)	23
Analyzed specimens (Windows executables)	14

Table 2. Results of Analysis

Observed Behavior	Counts
HTTP access to *Google*	4
NTP access to *time.windows.com*	2
Access to *windowsupdate.microsoft.com*	
DNS retrieve specific hosts	14

time.windows.com via NTP, were observed in some specimens. An unexpected behavior, an attempt to access to *windowsupdate.microsoft.com*, was also observed. In another behavior, which was expected and observed but not addressed, all specimens attempted to access each specific target. One expected behavior, attempts to probe network routes, was not observed. This shows that our mimetic Internet can fool malwares and that dynamically introducing targets is an important issue. It also shows, unexpectedly, that emulated routes are not important for current malwares.

We also showed that time intervals between each re-experiment were nearly zero because we prepared five nodes in the environment, and each re-experiment took 20 minutes. Because of this we were able to spend an hour rebuilding and an hour installing Windows XP, drivers, and tools. Thus, we can conclude that the node swapping method can reduce the time required to rebuild a virtualized environment.

7 Conclusion and Future Work

We focused on anti-analysis mechanisms of malwares and discussed how we can fool these mechanisms in an isolated sandbox. We used the mimetic Internet and the malware incubator with swappable actual nodes. We also conducted a proof-of-concept experiment on StarBED to test our proposed method and clarify issues. The results of the experiment show that the mimetic Internet is effective and the node swapping method is efficient, although some improvements must be made.

In the furure we plan to:

– Demonstrate that malwares that contain countermeasures against virtualized environments can be executed in our sandbox
– Implement an automated sandbox development and operation environment

- Design the dynamic introduction of specific target hosts for each malware to the mimetic Internet
- Design the mimetic dynamic contents generation on mimetic servers
- Enable specimens to access cryptographic services such as SSL
- Enable specimens to download files and join command through real networks from the isolated sandbox

We hope that by using our methods, malwares, which have anti-analysis mechanisms, such as the ability to detect isolated sandboxes and virtualized environments, will be analyzed correctly.

Acknowledgements

The authors give special thanks to Takashi UENO, a master's degree candidate at the Japan Advanced Institute of Science and Technology (JAIST) for his assistance in building, operating, and collecting specimens on a *nepenthes* environment.

References

1. Skoudis, E., Zeltser, L.: MALWARE – Fighting Malicious Code. Prentice Hall, Englewood Cliffs (2004)
2. MIWA, S., OHNO, H.: A Development of Experimental Environments "SIOS" and "VM Nebula" for Reproducing Internet Security Incidents. Journal of the National Institute of Information and Communications Technology 52(1/2), 23–34 (2005)
3. Eilam, E.: Reversing: Secrets of Reverse Engineering. Wiley Publishing, Chichester (2005)
4. The Honeynet Project, Know Your Enemy — Revealing the Security Tools, Tactics and Motives of the Blackhat Community. Addison Wesley, Reading (2002)(ISBN 0-201-74613-1, Pearson Education Corp.)
5. Smith, J.E., Nair, R.: Virtual Machines — Versatile Platforms for Systems and Processes. Morgan Kaufmann, San Francisco (2005)
6. King, S.T., Chen, P.M., Wang, Y., Verbowski, C., Wang, H.J., Lorch, J.R.: SubVirt: Implementing malware with virtual machines. In: Proceeding of IEEE Symposium on Security and Privacy (May 2006)
7. Introducing Blue Pill, http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html
8. Red Pill... or how to detect VMM using (almost) one CPU instruction, http://invisiblethings.org/papers/redpill.html
9. Blue Pill Detection! http://theinvisiblethings.blogspot.com/2006/08/blue-pill-detection.html
10. Wang, Y., Beck, D., Jiang, X., Roussev, R., Verbowski, C., Chen, S., King, S.: Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In: NDSS 2006. Proceedings of the Network and Distributed System Security Symposium, Internet Society 2006 (February 2006)

11. Vrable, M., Ma, J., Chen, J., Moore, D., Vandekieft, E., Snoeren, A.C., Voelker, G.M., Savage, S.: Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm. In: Proceedings of 20th ACM Symposium on Operating Systems Principles. SOSP 2005 (October 2005)
12. Crosby, S., Williams, D.E., Garcia, J.: Virtualization With Xen: Including XenEnterprise, XenServer, and XenExpress, Syngress Media Inc (2007), ISBN 1-597-49167-5
13. Miyachi, T., Chinen, K., Shinoda, Y.: StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software. In: Proceedings of First International Conference on Performance Evaluation Methodologies and Tools, Valuetools2006, ACM Press, New York (2006)
14. Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.: The Nepenthes Platform: An Efficient Approach to Collect Malware. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, Springer, Heidelberg (2006)
15. Kojm, T.: Clam AntiVirus, http://www.clamav.net/
16. Caswall, B., Beale, J., Foster, J.C., Faircloth, J.: Snort 2.0 Intrusion Detection. Syngress Publishing (2003), ISBN 1-9318-3674-4
17. Microsoft TechNet, Sysinternals Utilities: System Information, http://www.microsoft.com/technet/sysinternals/systeminformationutilities.mspx
18. Hemminger, S.: Network Emulation with NetEm. In: Proceedings of Linux Conf Au 2005 (April 2005)
19. INSECURE.ORG, Nmap - Free Security Scanner For Network Exploration & Security Audits, http://insecure.org/nmap/

Collusion-Resistant Fingerprinting Scheme Based on the CDMA-Technique

Naoki Hayashi[1], Minoru Kuribayashi[2], and Masakatu Morii[2]

[1] Graduate School of Science and Technology, Kobe University,
1-1, Rokkodai-cho, Nada-ku, Kobe, 657-8501 Japan
[2] Graduate School of Engineering, Kobe University,
1-1, Rokkodai-cho, Nada-ku, Kobe, 657-8501 Japan

Abstract. Digital fingerprinting is a scheme to insert user's own ID information into digital contents in order to identify illegal users who distribute unauthorized copy. One of the important problems is a collusion attack such that several users combine their copies of a same content to modify/delete the embedded fingerprint. In this paper, we propose a collusion-resilient fingerprinting scheme based on the CDMA technique. By combining spread spectrum sequence with orthogonal transform, we allocate the combination of spectrum components to users as their fingerprints. At a detector side, a threshold for the detection of colluded users is determined considering the false positive probability. We then estimate the parameters in the scheme for both optimal embedding and detection, and improve the accuracy of the detection of colluders. Experimental results show the effectiveness of proposed scheme.

1 Introduction

Accompanying with technology advancement, multimedia content (audio, image, video and etc.) has been closer to us and we have been able to easily access it. However, such an advantage also causes a serious problem that unauthorized users can duplicate digital content and redistribute it. In order to solve this problem, digital fingerprinting is used to trace back the illegal users, where unique IDs known as digital fingerprints [1], are embedded into content before distribution. When a suspicious copy is found, the owner can identify illegal users by extracting the fingerprint. A powerful cost-effective attack to the fingerprinting technique is a collusion such that several users with a same content that contains different fingerprints combine them together in order to generate a new version. One of the simplest approaches about the collusion attack is to average multiple copies of the content together. By combining many copies sufficiently, the fingerprints will be weakened or removed by the attack. Therefore, it is important to generate fingerprints that can be not only identify the colluders, but also resilient against collusion.

A number of works on designing fingerprints that are resistant against collusion attack have been proposed. Many of them can be categorized into two approaches. One is to devise an exclusive code, known as collusion-secure code

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 28–43, 2007.
© Springer-Verlag Berlin Heidelberg 2007

[3,4], which has traceability of colluders. However, making use of this code is not very practical since its code length is extremely long. The other is to exploit Spread Spectrum (SS) technique [2,5,6]. In the scheme, sequences which follow a normal distribution are assigned to each user as a fingerprint, and an excellent collusion resilience property is archived. Kirovski *et al.* proposed a dual watermark/fingerprint system that allows the use of different watermark signals in embedding and extraction [7], and an attack analysis and evaluation for the system are discussed in [8]. In [7], each user is supposed to have own media player that contains a watermark detector and its detection key which is sum of a common watermark and a fingerprint, and distributed contents are marked with the common watermark. When an attacker obtains the detection key from the player and subtracts the key from the watermarked content, not only will the watermark not be completely removed but also a fingerprint will be inserted in the attacked copy. Thus, our fingerprinting scheme different from their system with respect to the contents distribution model. The framework of SS embedding in [2] is extended using the coding technique [9,10]. However, several disadvantages are commonly remained in [2,7,9,10], such that high computational complexity is required in detection and a lot of fingerprints are needed for large number of users.

In this paper, we focus on the latter approach and propose a fingerprinting scheme based on the CDMA technique. The basic idea of our fingerprinting scheme is to allocate two spectrum components, which are orthogonal to the other components, and to express an unique ID as a fingerprinting by the combination of the components. In order to improve the allowable number of users in the system, we combine the quasi-orthogonal property of PN sequence [12] with orthogonal transform. Using this orthogonality, the inserted signals don't interfere with each other under averaging collusion. Therefore, it is possible to assign the combination of spectrum components to each user and to provide the hierarchical structure to the sequences. At a detection of the fingerprint information, we list up the components which signal energy exceeds a threshold, and identify the corresponding colluders whose fingerprints are expressed by the combination of the components. By applying the statistical property of the SS sequence, we give how to calculate a proper threshold according to the probability of false positive detection. Furthermore, we estimate the parameters used in the procedure of embedding and detection in the basic scheme. By giving a weight to the parameters, we improve the correct detection rate of colluders in the basic scheme. We demonstrate the performance of proposed scheme through computer simulation, in which the optimal parameters are estimated and their efficiency is clarified.

2 Preliminaries

2.1 Spread Spectrum Watermarking

A number of fingerprinting techniques have been proposed recently concerning to the robustness against collusion attack. On the technique, Cox *et al.*[2] first proposed watermarking scheme in which exploiting SS technique. In the work, SS sequence, which is assigned to each user as a fingerprint, consists of a sequence of real numbers, $X = \{x_1, \ldots, x_n\}$, where each element x_i is chosen independently according to $N(0, 1)$ (where $N(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2).

Let $V = \{v_1, \ldots, v_n\}$ be the frequency components of a digital image. We insert X into V to obtain a watermarked sequence V', for example, $v'_i = v_i + \alpha x_i$, where α is a embedding strength. At the detector side, we determine which SS sequence is present in a test image by evaluating the similarity of sequences. From the suspicious copy, a sequence X^* is detected by calculating the difference of the original image, and its similarity with X is obtained as follows.

$$\mathsf{sim}(X, X^*) = \frac{X \cdot X^*}{\sqrt{X^* \cdot X^*}}, \tag{1}$$

If the value exceeds a threshold, the embedded sequence is regarded as X.

In a fingerprinting scheme, each watermarked copy is slightly different, hence, malicious users will collect t copies $D_1, \ldots, D_t$ with respective watermark $X_1, \ldots, X_t$ in order to remove/alter the watermark. A simple, yet effective way is to average them because when t copies are averaged, $D^* = (D_1 + \ldots + D_t)/t$, the similarity value calculated by Eq. (1) results in shrinking by a factor of t, which will be roughly $\sqrt{n}/t$ [2]. Even in this case, we can detect the embedded watermark and identify the colluders by using an appropriately designed threshold.

This scheme has excellent robustness against signal processing, geometric distortions and subterfuge attacks [2]. However, it is not theoretically proven whether each SS sequence used in the scheme is (quasi-)orthogonal to each other. If some of them were correlated each other, the performance might be degraded. In order to get more robustness against averaging collusion, we have to use (quasi-)orthogonal sequences. In addition, this technique has a weakness such that the required number of SS sequences and computational complexity in detection increase with the number of users linearly. In our scheme, we exploit the sequences which consist of frequency components and allocate one of the spectrum components in the sequence to one user in order to let watermark be orthogonal to those for others. And, at the detection, we use only the difference between the test sequence and the original one without such a correlation detector as Eq. (1). So the computational complexity of detection is lower than the conventional scheme, and the preparation of the fingerprint sequences is not required.

2.2 Orthogonal Property

It is well-known that orthogonal transform such as DFT, DCT, etc. divides a sequence in mutually independent components like frequency elements. Since each spectrum component is orthogonal to the other components, changing only one component does not influence the other components. Considering its property, in our scheme, we allocate one of the components to the corresponding watermark information. For example, the components are denoted by a sequence $\boldsymbol{v} = \{v(0), \ldots, v(\ell-1)\}$. Then, we can insert one of ℓ kinds of watermarks to $\boldsymbol{v}$ by adding α to corresponding one component in $\boldsymbol{v}$, where α is an embedding strength. Since each component is mutually independent because of its orthogonality, we can assign it to each user as a fingerprint. In such a case, however, the allowable number of user is just that of components. In order to increase the number of user efficiently, it will be advisable to use two sequences; $\boldsymbol{v_0} = \{v_0(0), \ldots, v_0(\ell-1)\}$ and $\boldsymbol{v_1} = \{v_1(0), \ldots, v_1(\ell-1)\}$, and to allocate the combination of two components from different sequences to each user. In this case, ℓ^2 users can be admitted with 2ℓ spectrum components because the selection of two components has ℓ^2 candidates. However, under averaging attack, it causes a serious problem such that the combination of two components can't be identified uniquely even if we could detect some components containing watermark from test sequences. For example, we assign two components as a fingerprint to each user as shown in Table 1. If user 1 and user 6 collude to average two fingerprinted contents, then two components, $v_0(0)$ and $v_0(1)$ will be detected from $\boldsymbol{v_0}$, and similarly, two components, $v_1(0)$ and $v_1(2)$ from the other sequence $\boldsymbol{v_1}$ at a detector side. Here, even if we can detect such watermarked components, we can't identify the users uniquely since collusion pattern has two cases such as user 1 and user 6, or user 3 and user 4. In order to solve this problem, we exploit the quasi-orthogonal property of PN sequence such as M-sequence, Gold-sequence, etc [12]. It is noted that our method is similar to the CDMA technique for communication. Before embedding at the sequence $\boldsymbol{v_1}$, a specific PN sequence related to $\boldsymbol{v_0}$ is multiplied and then orthogonal transform is performed. As the results, each components in the obtained spectrum sequence is mutually independent, and if the applied PN sequence is different, the spectrum sequence is also mutually independent to other ones. Thus, we give hierarchical structure to the sequences for embedding, which increases the number of users; ℓ^2 users with only 2 ℓspectrum components. Then, we can identify colluders from the combination of detected IDs.

Table 1. An example of assigned fingerprint to 9 users

	$v_0(0)$	$v_0(1)$	$v_0(2)$
$v_1(0)$	user 1	user 4	user 7
$v_1(1)$	user 2	user 5	user 8
$v_1(2)$	user 3	user 6	user 9

In our technique, we suppose that each user's fingerprint consists of two parts of information. One is to identify the group where a user belongs to and the other represents an individual user in the group. Then, we call the group information "group ID" and the information for distinguishing individual users "user ID". The hierarchical structure in the sequences is illustrated in Fig.1. The details to embed and detect user's fingerprint are discussed in Section 3.

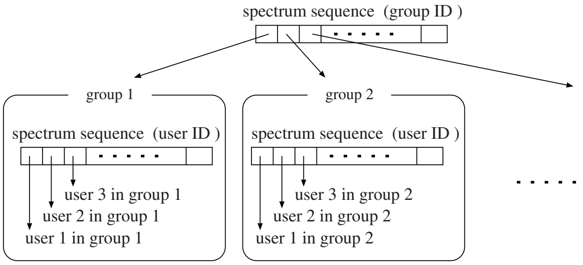


Fig. 1. Hierarchical structure of two sequences

2.3 Statistical Property

In conventional fingerprinting schemes [2,5], illegal users are detected by calculating correlations with original fingerprint. If an original data is available, the reliability of the detector can be increased. Here, it is strongly required for the detector to detect only illegal users, not innocent users. Therefore, the design of a threshold is to guarantee a probability of false positive detection. In this subsection, we exploit statistical properties to obtain the proper threshold for a given probability.

We call the sequence obtained by subtracting the host sequence from the watermarked sequence, "detection sequence", and it is denoted by $\boldsymbol{d} = \{d(0), \ldots, d(\ell-1)\}$. To apply statistical decision theory, we assume that $\boldsymbol{d}$ is composed of random variables and the sequence except watermarked component $d(k)$, which is denoted by $\boldsymbol{d}'$, are distributed according to $N(0, \sigma^2)$. If we insert a watermark to add α to $d(k)$ in order to satisfy $d(k) > \max_i\{d(i)\}(i \neq k)$, then we can detect the embedded watermark by setting a threshold T to be imposed $d(k) > T > d(i)$. Then, T can be put according with the probability of false detection, which is illustrated in Fig.2. The probability that a random variable $d(i)$ exceeds T, $Prob(d(i) > T)$, is equal to the marked area in Fig. 2. If $d(i) > T$, then a detector decides that $d(i)$ is watermarked and false detection occurs. Therefore, $Prob(d(i) > T)$ is the probability of false positive detection. We can, then, say that

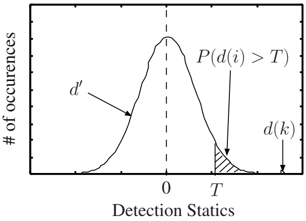


Fig. 2. Distribution of $\boldsymbol{d}'$ to be approximated to $N(0, \sigma^2)$

$$Prob(d(i) > T) \leq \frac{1}{2}\mathrm{erfc}\left(\frac{T}{\sqrt{2\sigma^2}}\right), \tag{2}$$

from study in [13]. By imposing, for example, that $Prob(d(i) > T) \leq 10^{-8}$, we find

$$T = 3.97\sqrt{2\sigma^2}. \tag{3}$$

If the variance σ^2 is known, we can obtain a proper threshold corresponding to a given probability of false detection.

3 Proposed Hierarchical Spread Spectrum Watermarking

In this section, we present our spread spectrum watermarking scheme. In our scheme, a fingerprint consists of two kinds of IDs, group ID and user ID, where group ID represents information of the group where a user belongs to and user ID is to distinguish individual user in the group, and each ID is embedded into one of the spectrum components extracted from host signal. In our embedding procedure, we realize the hierarchical structure between the sequences using PN sequence, which increase the number of users admitted in our fingerprinting system. Furthermore, assigning one spectrum component for embedding each ID, the watermark signal resists against averaging attack since the components forming the sequence are independent to each other. At a detection of the watermark, an optimal threshold is calculated for a given probability of false positive detection.

3.1 Approach

A fingerprint is embedded into the sequences which consists of the spectral components especially extracted from an host signal using a secret key. At inserting each ID, we multiply the sequence by PN sequence and further compute orthogonal transform of its sequence to obtain the orthogonal sequence for embedding. Each component in the sequence one-to-one corresponds to the position used for embedding the corresponding watermark and it is independent to each other. And each ID is embedded by adding α to the component which index is equal to each ID, where α is a watermark strength. Since the component is mutually independent from the other components, the watermarked signal dose not interfere with each other. Therefore, even when averaging attack is performed, a watermarked signal does not interfere with other watermarked ones, but each watermark energy is reduced by averaging. If the parameter α is properly designed, it is possible to immunize such an attack.

In addition, after adding α to the component, we compute inverse orthogonal transform to the watermarked sequence and multiply it by PN sequence. Then, embedded signal is spread over the sequence initially extracted from an host signal. Therefore, spreading the watermark throughout the former sequence ensures robustness against intentional or unintentional attacks. It is difficult to identify the location of the extracted because the embedded watermark is placed in the frequency domain. Only the owner knows the precise patterns of the watermark that is modulated by combination of DCT and PN sequence. In contrast, for an attacker the strategy to get the watermarked sequence is not realistic because he must know each parameters containing secret key used in the embedding procedure. Since an attacker only has knowledge of the possible range of modification, an attacker must create audible/visible defects in the data to eliminate a watermark. As other collusion attack, we assume that colluders obtain the difference by subtracting the watermarked signal from the other watermarked one, and exploit it to add noise to the watermarked signal in order to eliminate a watermark. However, since the additive noise is spread throughout the watermarked sequence by exploiting PN sequence, serious effects to a particular component in the watermarked sequence is difficult [5,10].

3.2 Embedding

We give the procedure to embed a user's fingerprint into $N \times N$ image. In our scheme, the allowable number of users is ℓ^2, and the fingerprint is denoted by (i_0, i_1), where i_0 represents group ID and i_1 represents user ID. The embedding procedure is summarized as follows.

1) Apply full-domain DCT to host image.
2) Extract 2ℓ DCT coefficients randomly from the low or middle frequency elements using a secret key, and create 2 sequences, $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ of length ℓ.
3) Embed i_0 and i_1 to $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$, respectively, by the following procedure.
   - group ID i_0
     a) Generate PN sequence $PN(s)$ of length ℓ using a secret key s as a seed, where the PN sequence consists of ±1. (Specifically, we use M-sequence in our scheme.)
     b) Multiply $\boldsymbol{v}_0$ by $PN(s)$ and apply 1- dimensional DCT (1-D DCT) to obtain an orthogonal sequence for embedding i_0, which is denoted by $\hat{\boldsymbol{v}_0} = \{\hat{v_0}(0), \hat{v_0}(1), \ldots, \hat{v_0}(\ell - 1)\}$.
     c) Add α_0 to $\hat{v_0}(i_0)$ to obtain $\hat{\boldsymbol{v}_0}'$.
     d) Apply 1-D IDCT of $\hat{\boldsymbol{v}_0}'$ and multiply it by $PN(s)$ to obtain a watermarked sequence $\boldsymbol{v}_0'$.
   - user ID i_1
     a) Generate PN sequence $PN(i_0)$ with length ℓ using i_0 as a seed.
     b) Multiply $\boldsymbol{v}_1$ by $PN(i_0)$ and apply 1-D DCT to obtain an orthogonal sequence for embedding i_1, which is denoted by $\hat{\boldsymbol{v}_1} = \{\hat{v_1}(0), \hat{v_1}(1), \ldots, \hat{v_1}(\ell - 1)\}$.
     c) Add α_1 to $\hat{v_1}(i_1)$ to obtain $\hat{\boldsymbol{v}_1}'$.
     d) Aplly 1-D IDCT of $\hat{\boldsymbol{v}_1}'$ and multiply it by $PN(i_0)$ to obtain a watermarked sequence $\boldsymbol{v}_1'$.
4) Substitute $\boldsymbol{v}_0'$ and $\boldsymbol{v}_1'$ for original DCT coefficients, and apply IDCT to obtain a watermarked image.

We add α_m to $\hat{v_m}(i_m), (m = 0, 1)$ in order to assign i_m-th spectrum component in $\hat{\boldsymbol{v}_m}$. Note that we have to decide a watermark strength α_m carefully since a larger

α_m increases the robustness against attacks but it also causes more degradation to the watermarked image. The detail of α_m is discussed in Section 4. After embedding i_m, we apply 1-D IDCT and multiply it by respective PN sequence, hence, the embedded watermark is spread over $\boldsymbol{v}_m{}'$. It is remarkable that the exploitation of $PN(i_0)$ enables us to make the sequence depended on the group ID i_0 for embedding user ID i_1, and to create the hierarchical structure among the sequences. Combination of the sequences can bring increase of allowable users comparing with the method based on one sequence.

3.3 Extraction

At the detector side, an host image and secret keys used in embedding are required. Since a user's fingerprint is embedded separately by two kinds of IDs, i_0 and i_1, a two-level detection algorithm is conducted. We first determine which group ID is present in a watermarked image to identify the group to which the illegal user belongs, and then we focus on the identified group to examine user ID. The latter operation is performed to the sequence using the PN sequence generated from the identified group ID as a seed. At the detection of each ID, we examine through comparing the components in detection sequence with a threshold. Here, we assume that two sequences extracted from a tested image is denoted by $\boldsymbol{v}_0^*$ and $\boldsymbol{v}_1^*$.

- group ID i_0
    1) Generate PN sequence $PN(s)$ of length ℓ using a secret key s as a seed.
    2) Subtract $\boldsymbol{v}_0$ from $\boldsymbol{v}_0^*$ to obtain the sequence $\boldsymbol{d}_0$.
    3) Multiply $\boldsymbol{d}_0$ by $PN(s)$ and apply 1-D DCT of it to obtain the detection sequence $\hat{\boldsymbol{d}}_0 = \{\hat{d}_0(0), \hat{d}_0(1), \ldots, \hat{d}_0(\ell-1)\}$.
    4) If $d_0(k)$ is more than a threshold T_0, detect k as group ID i_0.
- user ID i_1.
    1) Generate PN sequence $PN(i_0)$ of length ℓ using i_0 as a seed.
    2) Subtract $\boldsymbol{v}_1$ from $\boldsymbol{v}_1^*$ to obtain the sequence $\boldsymbol{d}_1$.
    3) Multiply $\boldsymbol{d}_1$ by $PN(i_0)$ and apply 1-D DCT to obtain the detection sequence $\hat{\boldsymbol{d}}_1 = \{\hat{d}_1(0), \hat{d}_1(1), \ldots, \hat{d}_1(\ell-1)\}$.
    4) If $d_1(k)$ is more than a threshold T_1, detect k as user ID i_1.

Note that when some group IDs are detected, we examine each user ID corresponding to each group ID in order to identify all colluders. Therefore, our scheme is designed for *catch many*-type fingerprinting [1]. Then The details of deciding the threshold T_0 and T_1 according to the probability of false detection is discussed in Section 4.

4 Optimization

In this section, we propose an optimal method to obtain a proper threshold and corresponding parameters. We first describe the specific technique in setting the threshold for examining the watermark, and then consider the parameters used in the fingerprinting scheme, estimating the parameters for optimal embedding and detection to our basic scheme presented in Section 3. The idea of our optimized scheme is to give weight to watermark strengths α_0 and α_1 at embedding two kinds of IDs and to also give bias to setting the corresponding thresholds T_0 and T_1 used in a two-level detection.

4.1 Threshold

In this subsection, we apply the statistical property discussed in subsection 2.3 for our basic scheme. In order to obtain a threshold which guarantees a given probability of false positive detection, we focus on the distribution of the detection sequence. Considering the property of the sequence, we obtain an approximation of variance σ^2 required for setting a threshold. In Fig. 3, for instance, we illustrate the detection sequence $\hat{\boldsymbol{d}}_0$ where group ID is embedded with the following conditions. For the adaption of Fast DCT, we choose $\ell = 2^{10} (= 1024)$. A watermark is embedded into different groups with a strength $\alpha_0 = \alpha_1 = 500$ in order to estimate the effects of averaging attack. For the evaluation of its practicality, we perform JPEG compression with quality 35% as well as the averaging attack. Fig. 3 depicts the detected signals from the attacked image, where the numbers in parenthesis represent group IDs. Each watermark energy is decreasing to $1/10$ of α_0 by averaging and additional noise interfered with both watermarked components and components which are not containing the watermark signal. However, 10 spikes which indicate the presence of the 10 group IDs. Then, the appropriately calculated threshold enables us to detect 10 groups where colluders come from. Further, we can similarly detect embedded user IDs, and finally identify the colluders. Note that the additional noise caused by JPEG compression shown in the non-watermarked components approximately follows a normal distribution. So we assume that the histogram of the signals in $\hat{\boldsymbol{d}}_0$ is illustrated in Fig. 4, where the histogram except for the watermarked signal is approximated to normal distribution. If we know σ^2 of the distribution of non-watermarked signals, then we can set the ideal threshold by using Eq. (2). In order to estimate σ^2, we focus on symmetric property of the distribution of non-watermarked components. Since we can easily choose the minimum component $\hat{d}_0(min)$ in $\hat{\boldsymbol{d}}_0$, the components, which are in the range D_0 from $\hat{d}_0(min)$ to $-\hat{d}_0(min)$, are assumed to non- watermarked signals. Similarly, for the detection sequence $\hat{\boldsymbol{d}}_1$, we can apply for these consideration as follows.

$$\sigma_m^2 = \frac{1}{n} \sum_{\hat{d_m}'(k) \in D_m} \left(\hat{d_m}'(k) - \overline{\hat{\boldsymbol{d}_m}'} \right)^2 , \tag{4}$$

where $\hat{\boldsymbol{d}_m}'$ denotes the detection sequence which components are in the range D_m at detecting each ID i_m, n denotes the number of components in $\hat{\boldsymbol{d}_m}'$, and $\overline{\hat{\boldsymbol{d}_m}'}$ is the mean of $\hat{\boldsymbol{d}_m}'$. Therefore, we can set a threshold according to the probability of false detection Pe_m.

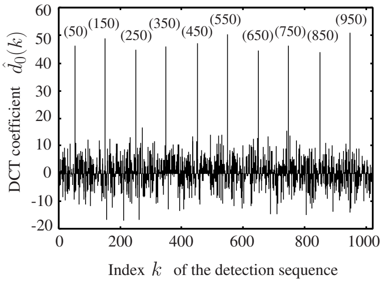


Fig. 3. The detected signals in the detection sequence $\hat{\boldsymbol{d}}_0$ under averaging attack and JPEG compression with quality 35%

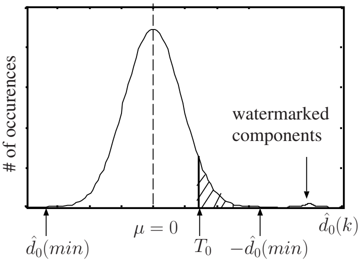


Fig. 4. The distribution of the detection sequence $\hat{\boldsymbol{d}}_0$ under averaging attack and JPEG compression with quality 35%

4.2 Weight

In this subsection, we consider the parameters in our scheme to improve the accuracy of detection of user's fingerprint under averaging attack. Our improved method is to give weight to the watermark strength α_m, and also to give weight to the probability Pe_m for setting the threshold T_m.

We first review the procedure to detect the watermark, in which a two-level detection scheme is conducted. At detection, after we detect group ID of each colluder, we detect each user ID which is corresponding to each group ID since group ID is necessary for generating the detection sequence to examine user ID in the group. Therefore, if we fail to detect the group ID in the first detection, the following procedure to detect user ID does not conducted, hence the probability of correct detection of user's fingerprint falls. In order to solve this problem, we provide weight to Pe_0 and Pe_1 which decide a threshold T_0 and T_1 respectively. By setting T_0 lower, the detection rate of group ID can be improved, however, the false positive detection rate will be also increased. Considering the false detection for detecting user ID, we put T_1 higher in order not to detect innocent user ID. Even if wrong group IDs are accidentally detected, user IDs associated with the wrong group IDs will be excluded with high probability. Thus, in our optimized scheme, we set $Pe_0 > Pe_1$ at the detecting procedure.

In our technique, we add a watermark with the strengths, α_0 and α_1 to each sequence for embedding. If the strengths are increased, the robustness against intentional or unintentional attack will be improved, but they also cause degradation. Hence, the watermark energy to be inserted is limited and we should apportion the limited energy between α_0 and α_1. If a strong energy is allocated to α_1, user IDs are more accurately detected. However, a larger α_1 also reduces the detection accuracy of group ID and makes it harder to narrow down the individual user in the group. From the above paragraph, a threshold T_0 should be lower in order to detect more group IDs of colluders even if wrong group IDs are detected. With a small α_0, we could expect to archive a maximum performance because a large α_1 brings the improvement on detection of user ID. Thus, we set

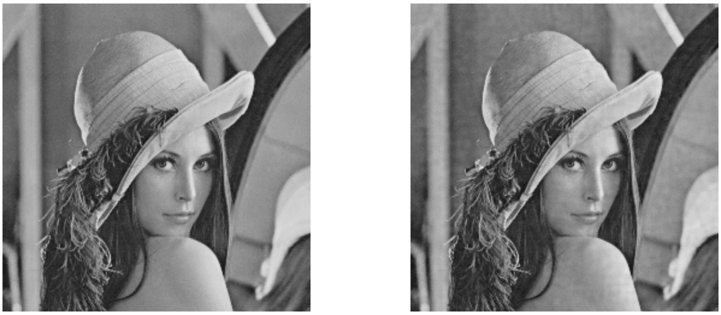
$\alpha_0 < \alpha_1$ at embedding procedure in our optimized fingerprinting scheme. The optimal parameters are estimated by computer simulation in Section 5.

5 Simulation Results

In this section, we implement the proposed scheme and evaluate the tolerance for collusion attack through computer simulation. We use the 256×256 "lenna" image as the host signals, and choose $\ell = 1024$, hence the total number of allowable users is $\ell^2 \simeq 10^6$. The detection of the fingerprint is performed with the knowledge of the host image.

5.1 Perceptual Quality

We evaluate the relation between a watermark strength and image quality. Considering that the degradation in the image will be noticeable when PSNR is less than 39 dB, we suppose that $\alpha_0 = \alpha_1 = 500$ are the maximum strength and assume them as the criterion of the image quality. Fig. 5 shows the original image and the watermarked image with 39.26 dB.



(a) Original image

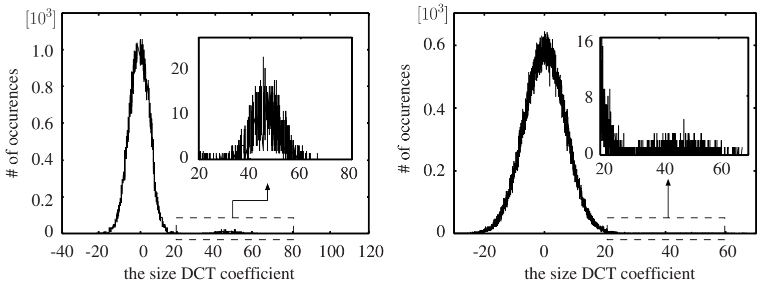
(b) Watermarked image $\alpha_0 = \alpha_1 = 500$

Fig. 5. Perceptual quality of "lenna"

5.2 Evaluation of Statistical Property

In our scheme, we set a threshold based on the statistical property of the distribution of each detection sequence. In order to examine the property, we first show the histogram of detection sequence $\hat{\boldsymbol{d}}_0$ and $\hat{\boldsymbol{d}}_1$ in Fig. 6 (a) and (b), respectively. In the simulation, watermarked images are averaged by 10 colluders with random groups and performed JPEG compression with quality 35 %. We set the watermark strength $\alpha_0 = \alpha_1 = 500$, and evaluate the performance using randomly selected 10^3 patterns of user ID. In Fig. 6 (a) and (b), it is confirmed that two distributions are contained; one is the distribution of non-watermarked and the other is that of watermarked components. The former distribution will follow a normal distribution with zero mean, and the mean value of the latter one will be 50 since watermarked signals are shrinking by a factor 10 from α_0 and α_1. When colluders belong to a small number of groups, the shrinking factor is decreased. For example, if 2 users out of 10 colluders belong to a same group, the corresponding watermarked component is distributed close to 100 in $\hat{\boldsymbol{d}}_0$.

Next, we examine whether the variance σ_m^2 calculated by Eq. (4) approximate to that of non-watermarked components, which is an ideal estimation. In a basic scheme, all components are used to calculate σ_m^2, which accuracy will be degraded by the watermarked components. In order to exclude such components efficiently, Eq. (4) is applied in our optimized scheme. The results are shown in Table 2. It is certified that the optimized scheme can obtain proper variances, hence thresholds can be designed based on the statistical property.



(a) Detection sequence $\hat{d}_0$. (b) Detection sequence $\hat{d}_1$.

Fig. 6. Distribution of detection sequences

Table 2. The estimation of variances

	group ID (σ_0^2)	user ID (σ_1^2)
non-watermarked	28.33	46.68
basic scheme	49.07	48.77
optimized scheme	28.09	46.24

5.3 Optimized Combination of Watermark Strengths

In the optimized scheme discussed in Section 4, we give weight to watermark strengths, α_0 and α_1. In order to obtain the optimum combination, we evaluate the detection rate of colluders for watermarked image with the watermark strength, which combination is decided with the PSNR about 39.26 dB considering $\alpha_0 = \alpha_1 = 500$. In the simulation, watermarked images are averaged by 10 colluders and performed JPEG compression with quality 35 %. We set $Pe_0 = 10^{-3}$ and $Pe_1 = 10^{-8}$, and use 10^4 patterns of colluders. The results are shown in Fig. 7. We observe from Fig. 7 that optimum α_0 is 370 and then α_1 is 605. In the following simulation, we use the combination as optimized scheme.

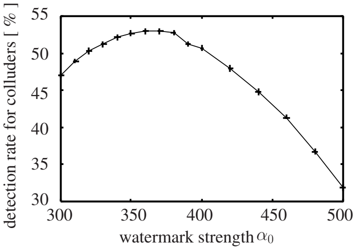


Fig. 7. Detection rate for colluders versus a watermark strength

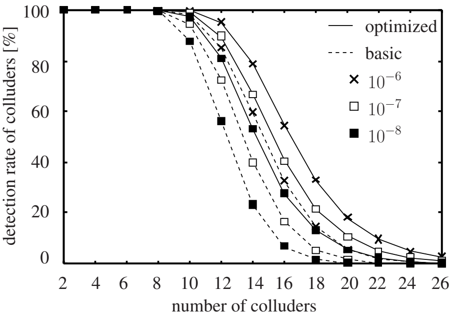


Fig. 8. Detection rate of colluders

5.4 Robustness Against Collusion

We evaluate the robustness against collusion attack of our scheme, and show the detection rate of colluders under averaging attack in Fig. 8. Watermarked images are averaged and performed JPEG compression with quality 35 %, and the simulation is iterated 10^4 times. We set $Pe_0 = 10^{-3}$ and $Pe_1 = 10^{-6} \sim 10^{-8}$. From Fig. 8, the detection rate of optimized scheme has up to approximately 20∼30% improvement over that of basic scheme when the number of colluders is from 12 to 18. In optimized scheme, If the number of colluders is less than 10, we can catch almost all colluders with very high probability. Fig. 9 shows the average number of false positive detection on the same conditions. Under less than 10 colluders for the optimized scheme, and 6 and 22 colluders for the basic scheme, the false positive detection does not occur. The false positive detection is slightly increased in the optimized scheme because the opportunity of detecting user ID is increased according to the detected number of group ID. Considering the improvement of the correct detection rate, however, such a slight shortcoming may be ignored.

We apply optimized scheme to the other images, “baboon”, “peppers” and “barbala” of the 256 × 256 pixels, and the detection rate of colluders is shown in Fig. 10. We set $Pe_0 = 10^{-3}$ and $Pe_1 = 10^{-8}$. We observe from Fig. 10 that there is no big difference on the performance of proposed scheme among the images, and we also confirm that the average number of false detection is close to that of

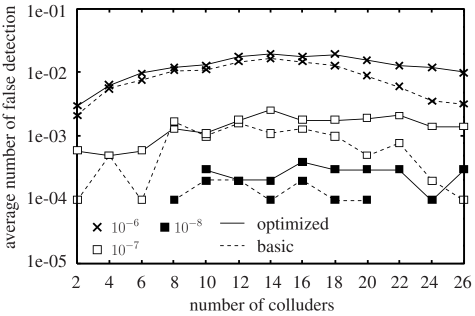


Fig. 9. Average number of false positive detection of optimized scheme and of basic scheme

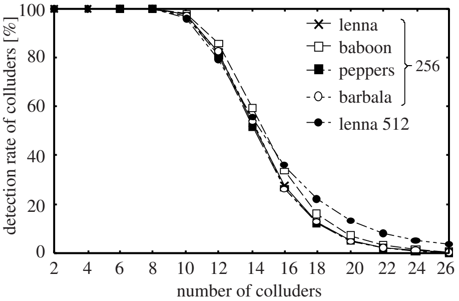


Fig. 10. Detection rate of colluders on the 256 × 256 images and the image "lenna" of 512 × 512

"lenna". We evaluate the performance of different size of images with a constant watermark energy. Fig. 10 also shows the detection rate on the image "lenna" of different size; 512 × 512 pixels, which is evaluated using randomly selected 10^3 patterns of colluders. We leave the consideration about the difference of performance between two images on future works.

5.5 Consideration

In the proposed scheme, 2ℓ DCT coefficients are used for the embedding of fingerprint information which is a pair (i_0, i_1). In order to realize the hierarchical structure, each ℓ coefficients are assigned for one of (i_0, i_1) without overlapping. Here, if each of them is multiplexed and embedded into 2ℓ coefficients based on CDMA technique, the hierarchical structure can also be realized. In such a case, the allowable number of users is $4\ell^2$, but the false positive detection rate will be doubled because the number of each elements are 2ℓ and the rate is proportionally increased by the number. In order to evaluate the positive detection rate under the same false detection rate as the proposed scheme, only half of

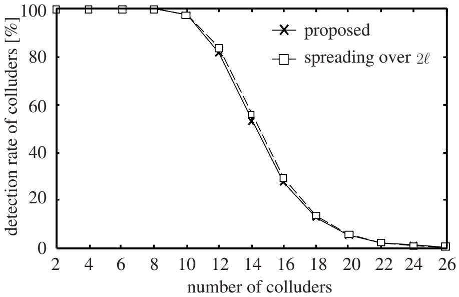


Fig. 11. Comparison the proposed scheme with spreading (i_0, i_1) over 2ℓ coefficients

the 1-D DCT coefficients computed from the 2ℓ coefficients are assigned for each of (i_0, i_1). Thus, the allowable number of users is also same. The comparison is depicted in Fig.11, and it indicates that spreading over 2ℓ coefficients based on CDMA technique superior. It is noted that the computational cost is also doubled since 1-D DCT is performed to 2ℓ coefficients.

6 Conclusion

In this paper, we proposed a collusion resilient fingerprinting scheme based on the CDMA technique. In the proposed scheme, each user's fingerprint consists of two information, group ID and user ID, and we assign these IDs to the combination of spectrum components. By exploiting the hierarchical structure provided from PN sequences, we can allow a larger number of users than conventional fingerprinting schemes. During fingerprint detection, we can put a threshold according to the probability of false positive detection. Since we do not calculate a similarity with a watermark, computational complexity is rather small. We then study the parameters in the scheme in order to obtain a maximum performance. By giving a weight to the probability for setting a threshold, we improved the correct detection rate of colluders. Using this improvement, moreover, we can give a weight to watermark strengths effectively, and improved the collusion resilience. We showed the effectiveness of the proposed scheme through experimental results.

References

1. Wu, M., Trappe, W., Wang, Z., Liu, K.J.R.: Collusion resistant fingerprinting for multimedia. IEEE Signal Processing Magazine, 15–27 (March 2004)
2. Cox, I., Kilian, J., Leighton, F., Shamson, T.: Secure spread spectrum watermarking for multimedia. IEEE Trans. on Image Process 6(5), 1673–1687 (1997)
3. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. IEEE Trans. Inform. Theory 44(5), 1897–1905 (1998)

4. Trappe, W., Wu, M., Wang, Z.J., Liu, K.J.R.: Anti-collusion fingerprinting for multimedia. IEEE Trans. Signal Process 51(4), 1069–1087 (2003)
5. Zhao, H., Wu, M., Wang, Z., Liu, K.J.R.: Forensic analysis of nonlinear collusion attacks for multimedia fingerprinting. IEEE Trans. on Image Process 14(5), 646–661 (2005)
6. Wang, Z.J., Wu, M., Zhao, H., Trappe, W., Liu, K.J.R.: Anti-collusion forensics of multimedia fingerprinting using orthogonal modulation. IEEE Trans. on Image Process 14(6), 804–821 (2005)
7. Kirovski, D., Malvar, H.S., Yacobi, Y.: Multimedia content screening using a dual watermarking and fingerprinting system. In: Proc. of ACM Multimedia, pp. 372–381 (2002)
8. Kesal, M., Mihcak, M.K., Venkatesan, R.: An improved attack analysis on a public-key spread spectrum watermarking. Journal of Multimedia systems 11(2), 133–142 (2005)
9. Zhu, Y., Feng, D., Zou, W.: Collusion secure convolutional spread spectrum fingerprinting. In: Barni, M., Cox, I., Kalker, T., Kim, H.J. (eds.) IWDW 2005. LNCS, vol. 3710, pp. 67–83. Springer, Heidelberg (2005)
10. He, S., Wu, M.: Joint coding and embedding techniques for multimedia fingerprinting. IEEE Trans. Inform. Forens 1(2), 231–247 (2006)
11. Fujitsu, S., Hagiwara, M., Imai, H., Kitagawa, T., Nuida, K., Ogawa, K., Watanabe, H.: Improved of 2-secure fingerprinting code's tracing algorithm. In: Proc. of SCIS2007, 3B1-1 (2007)
12. Gold, R.: Maximal recursive sequences with 3-valued recursive cross-correlation functions. IEEE Trans. Inform. Theory 14(1), 154–156 (1968)
13. Barni, M., Bartolini, F., Piva, A.: Improved wavelet-based watermarking through pixel-wise masking. IEEE Trans. on Image Process 10(5), 783–791 (2001)

Reduction Optimal Trinomials for Efficient Software Implementation of the η_T Pairing

Toshiya Nakajima[1], Tetsuya Izu[2], and Tsuyoshi Takagi[3]

[1] Fujitsu Ltd., 4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki, Kanagawa 211-8588, Japan
tossi@jp.fujitsu.com

[2] Fujitsu Laboratories Ltd., 4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki, Kanagawa 211-8588, Japan
izu@labs.fujitsu.com

[3] School of Systems Information Science, Future University-Hakodate,
116-2 Kamedanakano, Hakodate, Hokkaido 041-8655, Japan
takagi@fun.ac.jp

Abstract. The η_T pairing for supersingular elliptic curve over GF(3^m) has been paid attention because of its computational efficiency. Since most parts of computation of the η_T pairing are multiplications over GF(3^m), it is important to improve the speed of the multiplication when implementing the η_T pairing. In this paper we consider software implementation of multiplication over GF(3^m) and propose to use irreducible trinomials $x^m + ax^k + b$ over GF(3) such that w, bit length of word of targeted CPU, divides k. We call the trinomials "reduction optimal trinomials (ROTs)". ROTs actually exist for several m's and typical values of $w = 16$ and 32. We list them for extension degrees $m = 97$, 167, 193 and 239. These m's are derived from security considerations. Using ROT it is possible to implement efficient modulo operation (reduction) in multiplication over GF(3^m) comparing with the case using other type of trinomials (e.g., trinomials with minimum k for each m). The reason of this is that for the cases of reduction by ROT the number of shift operations on multiple precision data reduces to less than half comparing with the cases by other trinomials. Implementation results show that reduction algorithm specialized for ROT is 20–30% faster on 32-bit CPU and around 40% faster on 16-bit CPU than algorithm for irreducible trinomials with general k.

Keywords: pairing, software implementation, irreducible polynomial, reduction optimal trinomial, characteristic three, finite field.

1 Introduction

Recently pairings on elliptic curves are used as tools for cryptographic protocols. After firstly proposed to use pairings for ID-based non-interactive key sharing by Sakai, Ohgishi and Kasahara [18] and for tripartite generalization of Diffie-Hellman by Joux [13], many applications based on pairings have been proposed

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 44–57, 2007.
© Springer-Verlag Berlin Heidelberg 2007

such as ID-based cryptosystems by Boneh and Franklin [6], short signatures by Boneh, Lynn and Shacham [8], efficient broadcast cryptosystems by Boneh, Gentry and Waters [7], etc. Among two types of pairing, the Weil pairing and the Tate pairing, the Tate pairing is preferred for application because of its lower computational cost than that of the Weil pairing. However, since the cost of the Tate pairing is still high comparing with the cost of RSA cryptography for example, various techniques to realize fast computation of the Tate pairing were developed, and are still being developed.

Supersingular elliptic curves over finite field with characteristic three (GF(3^m)) for cryptographic applications are used by Koblitz [14], and by Galbraith [10]. The η_T pairing by Barreto, Galbraith, Ó' hÉigeartaigh and Scott [2] (also the Duursma-Lee algorithm by Duursma and Lee [9]) has been studied for speeding-up of the Tate pairing. It also uses supersingular elliptic curves over finite field with characteristic three and needs multiplication over GF(3^m) in most parts of the pairing computation. Thus it is important to improve the speed of the multiplication when implementing the η_T pairing.

In this paper we focus on a class of monic irreducible polynomials over GF(3) that realizes efficient modulo operation in multiplication over GF(3^m). We call the modulo operation reduction. For efficient reduction irreducible polynomials with minimum number of terms are preferable. It is easy to see that two-term irreducible polynomial $x^m + a$ over GF(3) for $m \geq 2$ is $x^2 + 1$ only, thus three-term irreducible polynomials (trinomials) $x^m + ax^k + b$ over GF(3) have been used if m allows their existence. However, it seems there are not so many published materials with arguments about choice of the degree k of the second term. Galbraith, Harrison and Soldera [11] shows irreducible polynomials used for implementation without stating the reason of choice. Bertoni, Guajardo, Kumar, Orlando, Paar and Wollinger [4] and Barreto [1] discuss their choices: Bertoni et al. used trinomials with $k < m/3$ for computation of cubing in GF(3^m). This choice in fact leads to search for irreducible trinomials with minimum k corresponding to m. Barreto pointed out that trinomials with $k \equiv m \pmod 3$ are useful for efficient computation of cube roots in GF(3^m). Since these choices are for cubing or cube root, the effectiveness of the choices depends on pairing algorithm used. Actually there is a version of the η_T pairing (and the Duursma-Lee algorithm) without cube roots and costs of cubings are rather small comparing with those of multiplications. Thus it would be useful to seek another method of choosing k, where main concern is to reduce the cost of multiplication.

Our Contributions:
We propose to use irreducible trinomials $x^m + ax^k + b$ over GF(3) such that w, bit length of word of targeted CPU (typical values of w are 16 and 32), divides k. We call the trinomials "reduction optimal trinomials (ROTs)". (The word "optimal" is used in the similar sense such as "optimal extension fields" or "optimal normal bases".) Using ROT we are able to reduce the number of shift operations in reduction to less than half comparing with the case to use trinomials with general (including minimum) k.

ROTs actually exist and not all k's are minimum for corresponding m's. We can determine them for extension degrees $m = 97$, 167, 193 and 239 with w=16 or 32. These m's are derived from security considerations of pairing cryptography without relation to k or w. Note that since reduction by ROT is in layer of finite field, our approach can be used simultaneously with many other speeding-up techniques being in higher layers for efficient pairing computation.

2 Preliminaries

2.1 The Tate Pairing

Let q be a power of a prime p and let E be an elliptic curve defined over finite field GF(q), then the Tate pairing is defined as a map

$$\langle\cdot,\cdot\rangle_l : \; E(\mathrm{GF}(q))[l] \times E(\mathrm{GF}(q^\kappa))/lE(\mathrm{GF}(q^\kappa)) \rightarrow \mathrm{GF}(q^\kappa)^*/(\mathrm{GF}(q^\kappa)^*)^l$$

where l is a positive integer s.t. $\gcd(l, q) = 1$ and $l | \#E(\mathrm{GF}(q))$. κ is called the embedding degree that is the smallest positive integer s.t. $l|(q^\kappa - 1)$. Although the image of the map is a quotient set, not an unique element in $\mathrm{GF}(q^\kappa)$, one can obtain unique value through operation called the final powering or the final exponentiation. This is called the reduced Tate pairing.

2.2 From Tate to η_T

The Miller algorithm by Miller [16] is the first algorithm that calculates the Tate pairing efficiently. Then improvements of performance of the algorithm have been achieved by many researchers. In this subsection we briefly follow some of these improvements.

The BKLS algorithm by Baretto, Kim, Lynn and Scott [3] uses supersingular elliptic curve $E^\beta : \; y^2 = x^3 - x + \beta \; ; \; \beta = \pm 1$ defined over finite field with characteristic three ($\mathrm{GF}(3^m)$) for efficient computation of the Tate pairing. The curve has property that tripling of a point $P = (x_p, y_p) \in E^\beta(\mathrm{GF}(3^m))$ is calculated simply as $[3]P = (x_p^9 - \beta, \; -y_p^9)$. This property leads to compute the Miller algorithm by the triple-and-add method that is faster than usual the double-and-add method.

The Duursma-Lee (DL) algorithm [9] chooses extension degree m of $\mathrm{GF}(3^m)$ such that $\gcd(m, 6) = 1$. In this case $\#E^\beta(\mathrm{GF}(3^{6m}))$ is $(3^{3m} + 1)^2$ and this reduces exponent of the final powering from $3^{6m} - 1$ to $3^{3m} - 1$. Furthermore, this algorithm computes the Miller algorithm in closed form with m iteration loops and needs no conditional branch inside the loop.

The Modified Duursma-Lee algorithm by Kwon [15] removes cube roots from the DL algorithm. This increases the number of cubings instead but in characteristic three cost of cubing is lower than that of cube root.

The η_T pairing [2] also has closed form of the Miller algorithm and the number of its iteration loops are almost half of those of the DL algorithm in characteristic three. Calculation cost inside the loop of the η_T pairing is nearly equal to that of the DL algorithm. On the other hand, exponent of the final powering of the η_T pairing is larger than that of the DL algorithm.

Beuchat, Shirase, Takagi and Okamoto [5] removes cube roots from the η_T pairing. Like as [15], this increases the number of cubings but reduces total cost of calculation of the η_T pairing. Moreover, Shirase, Takagi and Okamoto [19] improves performance of the final powering of the η_T pairing by utilizing the fact that part of the final powering is in torus $T_2(\mathrm{GF}(3^{3m}))$ and the fact that inverse of the torus is calculated with almost costless.

2.3 Algorithm of the η_T Pairing

The η_T pairing algorithm in characteristic three without cube roots [5] is shown in Algorithm 1 for the case of $m \equiv 1 (\mathrm{mod}\ 12)$. Since the embedding degree of the η_T pairing in characteristic three is six, Algorithm 1 shows that most of the calculations are performed over $\mathrm{GF}(3^m)$ and $\mathrm{GF}(3^{6m})$. Calculations over $\mathrm{GF}(3^{6m})$ can be performed over $\mathrm{GF}(3^m)$ by representing elements in $\mathrm{GF}(3^{6m})$ using basis of $\mathrm{GF}(3^{6m})/\mathrm{GF}(3^m)$. Therefore efficient implementation of $\mathrm{GF}(3^m)$ operations benefits for speeding-up of calculation of Algorithm 1. So we focus on calculations over $\mathrm{GF}(3^m)$ from now on, but before that we choose extension degree m from security considerations in the next subsection.

Algorithm 1. The η_T pairing (with final powering) for $m \equiv 1 \pmod{12}$ [5]

```
· Basis of $\mathrm{GF}(3^{6m})/\mathrm{GF}(3^m)$: $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$,   $\sigma^2 = -1,\ \rho^3 = \rho + \beta$
Input: $P = (x_p, y_p),\ Q = (x_q, y_q) \in E^{\beta}(\mathrm{GF}(3^m))[l]$
Output: $\eta_T(P, Q)$ with final powering
for $i = 1$ to $(m-1)/2$ do
    $x_p \leftarrow x_p^9 - \beta,\ y_p \leftarrow -y_p^9$ (in $\mathrm{GF}(3^m)$)
end for
if $\beta = 1$ then $y_p \leftarrow -y_p$ (in $\mathrm{GF}(3^m)$)
$d \leftarrow \beta$ (in $\mathrm{GF}(3)$)
$R_0 \leftarrow -y_p(x_p + x_q + \beta) + y_q\sigma + y_p\rho$ (in $\mathrm{GF}(3^{6m})$)
for $i = 0$ to $(m-1)/2$ do
    $r_0 \leftarrow x_p + x_q + d$ (in $\mathrm{GF}(3^m)$)
    $R_1 \leftarrow -r_0^2 + y_p y_q \sigma - r_0\rho - \rho^2$ (in $\mathrm{GF}(3^{6m})$)
    $R_0 \leftarrow (R_0 R_1)^3$ (in $\mathrm{GF}(3^{6m})$)
    $y_p \leftarrow -y_p,\ x_q \leftarrow x_q^9,\ y_q \leftarrow y_q^9$ (in $\mathrm{GF}(3^m)$)
    $d \leftarrow d - \beta$ (in $\mathrm{GF}(3)$)
end for
return $R_0^{(3^{3m}-1)(3^m+1)(3^m+1-\beta 3^{(m+1)/2})/3^m}$ (in $\mathrm{GF}(3^{6m})$)
```

2.4 Choice of Extension Degree m

The η_T pairing using supersingular elliptic curve E^{β} over $\mathrm{GF}(3^m)$ has embedding degree six that is maximum. Let r be the largest prime factor of order $\#E^{\beta}(\mathrm{GF}(3^m))$ and let $|\cdot|$ denote bit length, then to achieve 1024-bit RSA equivalent security level $|r|$ should be larger than around 160 bits and $|3^{6m}|$ should be larger than around 1024 bits. $|r|$ and $|3^{6m}|$ are called security parameters. If we need 2048-bit RSA equivalent security level, $|r|$ should be larger than around 224 bits and $|3^{6m}|$ should be larger than around 2048 bits. Since $\#E^{\beta}(\mathrm{GF}(3^m))$ is calculated as follows [2], we are able to determine extension degree m.

$$\#E^{\beta}(\mathrm{GF}(3^m)) = \begin{cases} 3^m + 1 + \beta 3^{(m+1)/2} & (m \equiv 1, 11 (\bmod 12)), \\ 3^m + 1 - \beta 3^{(m+1)/2} & (m \equiv 5, 7 \ (\bmod 12)). \end{cases}$$

Table 1 shows $|r|$ and $|3^{6m}|$ for several m's (with $\gcd(m, 6) = 1$). From this table we choose $m = 97$, $167(\beta{=}1)$, 193 and 239. Security levels are around 1024 bits for $m = 97$, and around 2048 bits for $m = 193$ and 239. $m = 167(\beta{=}1)$ has maximum security parameters between $m = 149$ and 187 in the table.

Table 1. Security parameters of E^{β}

m	β	$\|r\|$	$\|3^{6m}\|$
91	+1	116	866
97	+1	151	923
149	+1	220	1417
163	+1	256	1551
163	−1	259	1551
167	+1	262	1589
167	−1	237	1589
187	+1	245	1779
193	−1	306	1836
239	−1	379	2273

3 Multiplication over GF(3^m) and Reduction

For implementation of finite field operations, elements of finite field are represented by polynomial basis or normal basis. Granger, Page and Stam [12] compared both bases for software implementation of the DL algorithm and showed timing results that say pairing calculation by polynomial basis is much faster than those by normal basis. Since calculations inside the iteration loop of the DL algorithm are same as those of the η_T pairing (both are the cube roots removed versions), and since these loop calculations are dominant for total costs, we also adopt polynomial basis for software implementation of the η_T pairing.

Using polynomial basis, costs of addition and subtraction are negligible compared with multiplication. No division or inversion is needed inside the iteration loop of the η_T pairing. Multiplication over GF(3^m) with polynomial basis consists of two kinds of operations: (1) multiplication of two polynomials over GF(3) and (2) modulo operation by degree m irreducible polynomial over GF(3). We call this modulo operation reduction. Our aim in this paper is to choose suitable irreducible polynomials for efficient software implementation of reduction.

4 Reduction by Trinomial

Irreducible polynomials for efficient reduction should have minimum number of terms as long as such polynomials exist. Since x^2+1 is the unique two-term irreducible polynomial over GF(3) for $m \geq 2$, we focus on three-term polynomials (trinomials) $x^m + ax^k + b$ $(m > k > 0;\ a, b \in$GF$(3)^*)$. Reduction algorithm by irreducible trinomial is shown in Algorithm 2. Since m is already determined in subsection 2.4 by security considerations of pairing-based cryptography, we have to choose rest parameters k, a, b for efficient implementation of Algorithm 2.

Algorithm 2. Reduction by trinomial

```
Input: Degree $n(> m-1)$ polynomial over GF(3) $C(x) = \sum_{i=0}^{n} c_i x^i$C
       Irreducible trinomial over GF(3) $p(x) = x^m + ax^k + b$.
Output: $C(x) \leftarrow C(x) \bmod p(x)$
for $i = n$ downto $m$ do
    $c_{i-m+k} \leftarrow c_{i-m+k} - ac_i$
    $c_{i-m} \leftarrow c_{i-m} - bc_i$
    $c_i \leftarrow 0$
end for
```

4.1 Implementation of Reduction Algorithm

When applying Algorithm 2 to software implementation for general-purpose CPUs (including CPUs for embedding systems), it is efficient to process multiple number of coefficients of degree n input polynomial simultaneously. The number is width (bit length) of word of targeted CPU. We assume this number w is equal to bit length of CPU's general-purpose registers.

Since c_i of Algorithm 2 is in GF(3), we store each coefficients by bit-slicing as mentioned in Page and Smart [17], that is, higher bits and lower bits of w coefficients are stored separately in two array elements with bit length w. Using two arrays H and L, each has $M := \lceil (n+1)/w \rceil$ elements, we store $c_0 = cH_0||cL_0, \ldots, c_n = cH_n||cL_n$ as follows:

$$\begin{aligned} H[M-1] &= 0||\ldots||0||cH_n||\ldots||cH_{n+1-\{(n+1) \bmod w\}}, \\ L[M-1] &= 0||\ldots||0||cL_n||\ldots||cL_{n+1-\{(n+1) \bmod w\}}, \\ &\ldots \\ H[0] &= cH_{w-1}||\ldots||cH_0, \\ L[0] &= cL_{w-1}||\ldots||cL_0. \end{aligned}$$

Where cH_i and $cL_i (i = 0, \ldots, n)$ are higher and lower bits of c_i, respectively, and $||$ is bit concatenation. Then we are able to calculate w operations over GF(3) simultaneously like a SIMD instruction.

Reduction scheme and algorithm with above data structure are shown in Figure 1 and Algorithm 3, respectively. We restrict degree of input polynomial

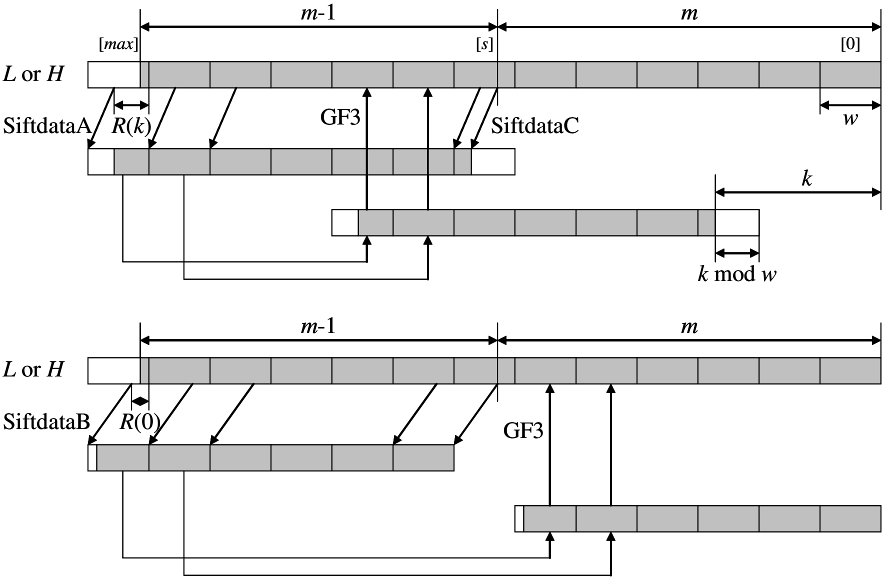


Fig. 1. Reduction scheme

is at most $2m - 2$ (the number of coefficients is at most $2m - 1$). Algorithm 3 is for irreducible trinomials over GF(3) with arbitrary parameters k, a, b.

Functions ShiftdataA, ShiftdataB, ShiftdataC and GF3 in Algorithm 3 are described as follows:

First, we define a function $R(k)$ that determines word boundary of shifted data.

$$R(k) := [\{\lceil\{(m-1) + (k \bmod w)\}/w\rceil\}w - (m-1) - (k \bmod w)] \\ + \{(2m-1) \bmod w\}.$$

Then the four functions in Algorithm 3 are as follows:

ShiftdataA(X, Y, m, k):
Returns shifted data corresponding to step 2 in Algorithm 2:
($R(k)$ bits from LSB of X)||($w - R(k)$ bits from MSB of Y).

ShiftdataB(X, Y, m):
Returns shifted data corresponding to step 3 in Algorithm 2:
($R(0)$ bits from LSB of X)||($w - R(0)$ bits from MSB of Y).

ShiftdataC($X[s], m, k$):
Returns last shifted data corresponding to step 2 in Algorithm 2:
($w - (k \bmod w)$ bits from ($m \bmod w$)-th bit (LSB=0th) of $X[s]$)||
(($k \bmod w$) bits of 0).

Algorithm 3. Reduction with word-wise data

```
Input:
  Array $H, L$ that store higher and lower bits of coefficients of degree $2m - 2$
  polynomial, respectively,
  Degree $m$ irreducible trinomial $p(x) = x^m + ax^k + b$,
  Width of word $w$ in bits,
  Maximum index of array $max = \lceil (2m - 1)/w \rceil - 1$,
  Index $s$. $H[s]$ and $L[s]$ store $cH_m$ and $cL_m$, respectively.
Output:
  Array $H, L$ that reduction results are stored in.
Allocate unsigned one-word length variables $th, tl$
$j_1 \leftarrow \lceil \{(m - 1) + (k \bmod w)\}/w \rceil$
$j_2 \leftarrow \lceil (m - 1)/w \rceil$
for $i = max$ downto $s + 1$ do
    $th \leftarrow$ ShiftdataA$(H[i], H[i - 1], m, k)$
    $tl \leftarrow$ ShiftdataA$(L[i], L[i - 1], m, k)$
    $j_1 \leftarrow j_1 - 1$
    $j_3 \leftarrow j_1 + \lfloor k/w \rfloor$
    GF3$(H[j_3], L[j_3], th, tl, a)$
    $th \leftarrow$ ShiftdataB$(H[i], H[i - 1], m)$
    $tl \leftarrow$ ShiftdataB$(L[i], L[i - 1], m)$
    $j_2 \leftarrow j_2 - 1$
    GF3$(H[j_2], L[j_2], th, tl, b)$
    $H[i] \leftarrow 0, L[i] \leftarrow 0$
end for
$th \leftarrow$ ShiftdataC$(H[s], m, k)$
$tl \leftarrow$ ShiftdataC$(L[s], m, k)$
$j_3 \leftarrow \lfloor k/w \rfloor$
GF3$(H[j_3], L[j_3], th, tl, a)$
All $w - (m \bmod w)$ bits from MSB of $H[s] \leftarrow 0$
All $w - (m \bmod w)$ bits from MSB of $L[s] \leftarrow 0$
```

GF3$(H[j], L[j], th, tl, X)$:
Stores higher and lower bits of GF(3) operation $A - XB$ in $H[j]$ and $L[j]$, respectively. Where $A := H[j]_z || L[j]_z$, $B := th_z || tl_z$; and suffix z is z-th bit from LSB $(0 \leq z \leq w - 1)$.

Note that there is another case of $R(k)$ that the last term of its RHS is replaced with $-[w - \{(2m - 1) \bmod w\}]$. In this case Algorithm 3 is modified slightly but we omit the detail to keep arguments clear.

5 Proposed Method for Choice of Degree *k*

For efficient software implementation of reduction, we propose to choose degree k of irreducible trinomial $x^m + ax^k + b$ over GF(3) as $w|k$ where w is bit length of word of targeted CPU. We call such irreducible trinomial "reduction optimal trinomial (ROT)".

5.1 Existence of ROTs

There are 320 ROTs with $8|k$ for $m \leq 256$. List of them is shown in Appendix. We found these trinomials using Risa/Asir, an open source computer algebra system [21]. Among these 320 ROTs there are 108 ROTs with $\gcd(m, 6) = 1$.

For typical values of $w = 16$ and 32, ROTs corresponding with predetermined $m = 97$, 167, 193 and 239 (subsection 2.4) do exist:

$$\begin{aligned}
x^{97} + ax^{16} + b&,\\
x^{167} + ax^{96} + b &= x^{167} + ax^{3\cdot 32} + b,\\
x^{193} + ax^{64} + b &= x^{193} + ax^{2\cdot 32} + b,\\
x^{193} + ax^{112} + b &= x^{193} + ax^{7\cdot 16} + b,\\
x^{239} + ax^{96} + b &= x^{239} + ax^{3\cdot 32} + b.
\end{aligned}$$

Where $(a, b) = (1, 2)$ or $(2, 1)$.

Note that if m is odd and k is even, irreducibility of $x^m + x^k + 2$ coincides with that of $x^m + 2x^k + 1$ because $x^m + x^k + 2 = 2(2x)^m + (2x)^k + 2 = 2(x^m + 2x^k + 1)$. A necessary condition for irreducibility of a trinomial over GF(3) is given by von zur Gathen [20].

5.2 Efficient Implementation of Reduction Algorithm

Using ROT we are able to improve Algorithm 3 because in this case we have same *th* (step 4 and 9), and same *tl* (step 5 and 10) by function ShiftdataB. Moreover, actual data in *th* and *tl* are word-aligned thus last call of function GF3 (step 18) can be omitted (choice of $(a, b) = (1, 2)$ or $(2, 1)$ does not affect execution time of function GF3). Figure 2 and Algorithm 4 show improved version of Figure 1 and Algorithm 3, respectively.

5.3 Comparison of Function Calls in Algorithm 4 and 3

We compare efficiency of Algorithm 4 that uses ROT with efficiency of Algorithm 3 that uses other type of irreducible trinomial by counting function calls in each algorithm. The ROTs we concern are with (m, k) = (97,16), (167,96), (193,64) and (239,96) corresponding with w = 16, 32, 32 and 32, respectively, while other irreducible trinomials are with (m, k) = (97,12), (167,92), (193,12) and (239,24), i.e., each k is minimum for corresponding m. We call three functions ShiftdataA, ShiftdataB and ShiftdataC simply “shift” and denote by S. We also call function GF3 simply “addition” and denote by A. Counting the function calls for $m = 97$, there are 26 shifts and 13 additions in Algorithm 3 while in Algorithm 4 the number of shifts are reduced to less than half: 12 shifts and 12 additions. This result is similar for other cases of m as shown in Table 2.

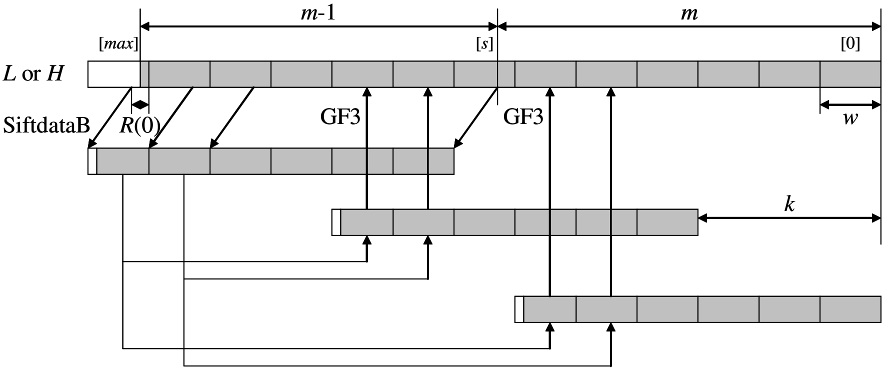


Fig. 2. Reduction scheme by ROT

Algorithm 4. Reduction by ROT

```
Input and Output: Same as Algorithm 3.
Allocate unsigned one-word length variables $th, tl$
$j_2 \leftarrow \lceil (m-1)/w \rceil$
for $i = max$ downto $s+1$ do
    $th \leftarrow$ ShiftdataB$(H[i], H[i-1], m)$
    $tl \leftarrow$ ShiftdataB$(L[i], L[i-1], m)$
    $j_2 \leftarrow j_2 - 1$
    $j_3 \leftarrow j_2 + \lfloor k/w \rfloor$
    GF3$(H[j_3], L[j_3], th, tl, a)$
    GF3$(H[j_2], L[j_2], th, tl, b)$
    $H[i] \leftarrow 0, L[i] \leftarrow 0$
end for
All $w - (m \bmod w)$ bits from MSB of $H[s] \leftarrow 0$
All $w - (m \bmod w)$ bits from MSB of $L[s] \leftarrow 0$
```

Table 2. Number of shifts and additions

m	Algorithm 4	Algorithm 3
97	12S+12A ($k = 16$)	26S+13A ($k = 12$)
167	12S+12A ($k = 96$)	26S+13A ($k = 92$)
193	12S+12A ($k = 64$)	26S+13A ($k = 12$)
239	16S+16A ($k = 96$)	34S+17A ($k = 24$)

S: shift, A: addition

6 Implementation Results

We compare timings of software implementations of Algorithm 4 and 3. Irreducible trinomials are same as those in subsection 5.3. Coding is in C language and targeted CPUs are Fujitsu's 32-bit RISC [22] and 16-bit CISC [23] that are for embedded systems. In the implementation for $m = 97$, 16-bit unsigned variables are used and for other m's 32-bit unsigned variables are used. Timings are measured in clock cycles using the simulator for the CPUs. Optimizations for the C code are loop unrolling and function inlining. Tunings by hand in assembly language are not applied.

Table 3 shows the timing results of the implementation in clock cycles for each m and CPU. Ratios of clock cycles in Algorithm 4 to those in Algorithm 3 are also shown. We can see from Table 3 Algorithm 4 is about 20–30% faster than Algorithm 3 for 32-bit CPU, and about 40% faster for 16-bit CPU (The increase of clock cycles in the case of $m = 97$ for 32-bit CPU is because 16-bit variables were used). The reason of the speeding-up is that the decrease of the number of shifts in Algorithm 4 results in less use of high-cost load/store instructions between the register and the memory.

Table 3. Implementation results

	32-bit CPU			16-bit CPU		
m	Algorithm 4	Algorithm 3	ratio	Algorithm 4	Algorithm 3	ratio
97	562cc	720cc	**0.78**	1180cc	1929cc	**0.61**
167	253	361	**0.70**	1884	3008	**0.63**
193	265	375	**0.71**	1996	3234	**0.62**
239	340	477	**0.71**	2556	4057	**0.63**

(cc: clock cycles)

7 Summary and Conclusions

We proposed to use reduction optimal trinomials (ROTs) for efficient software implementation of reduction in multiplication over GF(3^m). An ROT is an irreducible trinomial $x^m + ax^k + b$ over GF(3) with property that w (bit length of word of targeted CPU) divides k. Using ROTs for reduction it is able to reduce the number of shift operations to less than half comparing with the number using trinomials with other k's. For predetermined m's that are derived from security considerations of pairing-based cryptography, ROTs exist for typical values of $w = 16$ ($m = 97$, 167, 193 and 239) and $w = 32$ ($m = 167$, 193 and 239). Implementation results show that reductions by ROTs are about 20–30% faster for 32-bit CPU and about 40% faster for 16-bit CPU compared with reductions by trinomials with minimum k for each m. Since reduction by ROT is in layer of finite field, our approach can be used simultaneously with many other speeding-up techniques being in higher layers for efficient pairing computation. Future work will be to extend the concept of ROT to irreducible polynomials with four or more terms in the case of degree m with which no ROTs exist.

Acknowledgements

The authors thank Dr. Masaaki Shirase at Future University-Hakodate for his useful discussions with us. The authors also thank anonymous referees for their valuable comments.

References

1. Barreto, P.S.L.M.: A Note on Efficient Computation of Cube Roots in Characteristic Three, Cryptology ePrint Archive, Report 2004/305 (2004)
2. Barreto, P.S.L.M., Galbraith, S.D., Ó'hÉigeartaigh, C., Scott, M.: Efficient Pairing Computation on Supersingular Abelian Varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007) Designs
3. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
4. Bertoni, G., Guajardo, J., Kumar, S., Orlando, G., Paar, C., Wollinger, T.: Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 158–175. Springer, Heidelberg (2003)
5. Beuchat, J.-L., Shirase, M., Takagi, T., Okamoto, E.: An Algorithm for the η_T Pairing Calculation in Characteristic Three and its Hardware Implementation. In:18th IEEE International Symposium on Computer Arithmetic, ARITH-18, pp.97–104 (2007) In:18th IEEE International Symposium on Computer Arithmetic
6. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boneh, D., Gentry, C., Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
8. Boneh, D., Lynn, B., Shacham, H.: Short Signature from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
9. Duursma, I., Lee, H.-S.: Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$. In: Laih, C S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
10. Galbraith, S.D.: Supersingular Curves in Cryptography. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 495–513. Springer, Heidelberg (2001)
11. Galbraith, S.D., Harrison, K., Soldera, D.: Implementing the Tate Pairing. In: Fieker, C., Kohel, D.R. (eds.) ANTS-V 2002. LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
12. Granger, R., Page, D., Stam, M.: Hardware and Software Normal Basis Arithmetic for Pairing–Based Cryptography in Characteristic Three. IEEE Trans. on Computers 54(7), 852–860 (2005)
13. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS-IV. LNCS, vol. 1838, pp. 385–393. Springer, Heidelberg (2000)
14. Koblitz, N.: An Elliptic Curve Implementation of the Finite Field Digital Signature Algorithm. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 327–337. Springer, Heidelberg (1998)
15. Kwon, S.: Efficient Tate Pairing Computation for Supersingular Elliptic Curves over Binary Fields. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 134–145. Springer, Heidelberg (2005)
16. Miller, V.S.: Short Programs for Functions on Curves, (Unpublished manuscript), Available at http://crypto.stanford.edu/miller/miller.pdf Short Programs for Functions on Curves
17. Page, D., Smart, N.P.: Hardware Implementation of Finite Fields of Characteristic Three. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 529–539. Springer, Heidelberg (2003)

18. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems Based on Pairing, in Proceedings of the 2000 Symposium on Cryptography and Information Security, SCIS2000-C70 (2000) Cryptosystems Based on Pairing, in Proceedings of the 2000 Symposium on Cryptography and Information Security, SCIS2000-C70
19. Shirase, M., Takagi, T., Okamoto, E.: Some Efficient Algorithms for the Final Exponentiation of η_T Pairing, 3rd Information Security Practice and Experience Conference. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 254–268. Springer, Heidelberg (2007)
20. von zur Gathen, J.: Irreducible Trinomials over Finite Fields. Math. Comp. 72(244), 1987–2000 (2003) Irreducible Trinomials over Finite Fields. Math. Comp.
21. Risa/Asir: Available at http://www.math.kobe-u.ac.jp/Asir/asir.html
22. Document of 32-bit CPU: Available at http://www.fujitsu.com/us/services/edevices/microelectronics/microcontrollers/fr32bitseries/
23. Document of 16-bit CPU: Available at http://www.fujitsu.com/us/services/edevices/microelectronics/microcontrollers/f2mc/16lcore.html

Appendix: Irreducible trinomials with $8|k$ for $m \leq 256$

$x^{11} + x^{8} + 2$
$x^{15} + x^{8} + 2$
$x^{16} + x^{8} + 2$
$x^{17} + x^{16} + 2$
$x^{19} + x^{8} + 2$
$x^{21} + x^{16} + 2$
$x^{23} + x^{8} + 2$
$x^{32} + x^{8} + 2$
$x^{32} + x^{16} + 2$
$x^{32} + x^{24} + 2$
$x^{37} + x^{24} + 2$
$x^{39} + x^{32} + 2$
$x^{41} + x^{40} + 2$
$x^{47} + x^{32} + 2$
$x^{48} + x^{8} + 2$
$x^{48} + x^{40} + 2$
$x^{53} + x^{40} + 2$
$x^{55} + x^{32} + 2$
$x^{64} + x^{16} + 2$
$x^{64} + x^{24} + 2$
$x^{64} + x^{32} + 2$
$x^{64} + x^{40} + 2$
$x^{64} + x^{48} + 2$
$x^{67} + x^{56} + 2$
$x^{73} + x^{72} + 2$
$x^{77} + x^{16} + 2$
$x^{81} + x^{40} + 2$
$x^{83} + x^{32} + 2$
$x^{83} + x^{56} + 2$
$x^{85} + x^{16} + 2$
$x^{89} + x^{64} + 2$
$x^{95} + x^{48} + 2$
$x^{96} + x^{16} + 2$
$x^{96} + x^{80} + 2$
$x^{97} + x^{16} + 2$
$x^{99} + x^{80} + 2$
$x^{103} + x^{56} + 2$
$x^{107} + x^{32} + 2$
$x^{107} + x^{96} + 2$
$x^{107} + x^{104} + 2$
$x^{109} + x^{88} + 2$
$x^{109} + x^{96} + 2$
$x^{112} + x^{8} + 2$
$x^{112} + x^{104} + 2$
$x^{115} + x^{32} + 2$
$x^{121} + x^{40} + 2$
$x^{121} + x^{120} + 2$
$x^{127} + x^{8} + 2$
$x^{128} + x^{32} + 2$
$x^{128} + x^{48} + 2$
$x^{128} + x^{56} + 2$
$x^{128} + x^{64} + 2$
$x^{128} + x^{72} + 2$
$x^{128} + x^{80} + 2$
$x^{128} + x^{96} + 2$
$x^{131} + x^{48} + 2$
$x^{131} + x^{104} + 2$
$x^{133} + x^{88} + 2$
$x^{137} + x^{136} + 2$
$x^{139} + x^{80} + 2$
$x^{141} + x^{64} + 2$
$x^{141} + x^{136} + 2$
$x^{144} + x^{56} + 2$
$x^{144} + x^{88} + 2$
$x^{145} + x^{24} + 2$
$x^{145} + x^{64} + 2$
$x^{145} + x^{72} + 2$
$x^{147} + x^{8} + 2$
$x^{147} + x^{104} + 2$
$x^{157} + x^{88} + 2$
$x^{159} + x^{32} + 2$
$x^{160} + x^{40} + 2$
$x^{160} + x^{120} + 2$
$x^{163} + x^{80} + 2$
$x^{163} + x^{104} + 2$
$x^{165} + x^{88} + 2$
$x^{167} + x^{96} + 2$
$x^{169} + x^{24} + 2$
$x^{169} + x^{64} + 2$
$x^{176} + x^{40} + 2$
$x^{176} + x^{136} + 2$
$x^{179} + x^{104} + 2$
$x^{179} + x^{120} + 2$
$x^{181} + x^{40} + 2$
$x^{181} + x^{144} + 2$
$x^{185} + x^{64} + 2$
$x^{187} + x^{8} + 2$
$x^{187} + x^{32} + 2$
$x^{187} + x^{56} + 2$
$x^{191} + x^{120} + 2$
$x^{192} + x^{32} + 2$
$x^{192} + x^{160} + 2$
$x^{193} + x^{64} + 2$
$x^{193} + x^{112} + 2$
$x^{195} + x^{104} + 2$
$x^{201} + x^{88} + 2$
$x^{203} + x^{8} + 2$
$x^{203} + x^{32} + 2$
$x^{203} + x^{200} + 2$
$x^{205} + x^{144} + 2$
$x^{205} + x^{168} + 2$
$x^{208} + x^{56} + 2$
$x^{208} + x^{152} + 2$
$x^{209} + x^{40} + 2$
$x^{209} + x^{160} + 2$
$x^{224} + x^{16} + 2$
$x^{224} + x^{104} + 2$
$x^{224} + x^{120} + 2$
$x^{224} + x^{208} + 2$
$x^{225} + x^{16} + 2$
$x^{227} + x^{120} + 2$
$x^{227} + x^{216} + 2$
$x^{229} + x^{72} + 2$
$x^{235} + x^{152} + 2$
$x^{239} + x^{24} + 2$
$x^{239} + x^{56} + 2$
$x^{239} + x^{96} + 2$
$x^{239} + x^{104} + 2$
$x^{240} + x^{8} + 2$
$x^{240} + x^{232} + 2$
$x^{241} + x^{88} + 2$
$x^{253} + x^{136} + 2$
$x^{253} + x^{184} + 2$
$x^{256} + x^{40} + 2$
$x^{256} + x^{64} + 2$
$x^{256} + x^{96} + 2$
$x^{256} + x^{112} + 2$
$x^{256} + x^{128} + 2$
$x^{256} + x^{144} + 2$
$x^{256} + x^{160} + 2$
$x^{256} + x^{192} + 2$
$x^{256} + x^{216} + 2$
$x^{10} + 2x^{8} + 1$
$x^{11} + 2x^{8} + 1$
$x^{15} + 2x^{8} + 1$
$x^{17} + 2x^{16} + 1$
$x^{18} + 2x^{8} + 1$
$x^{19} + 2x^{8} + 1$
$x^{21} + 2x^{16} + 1$
$x^{22} + 2x^{16} + 1$
$x^{23} + 2x^{8} + 1$
$x^{26} + 2x^{8} + 1$
$x^{26} + 2x^{24} + 1$
$x^{30} + 2x^{16} + 1$
$x^{34} + 2x^{32} + 1$
$x^{37} + 2x^{24} + 1$
$x^{38} + 2x^{16} + 1$
$x^{39} + 2x^{32} + 1$
$x^{41} + 2x^{40} + 1$
$x^{42} + 2x^{32} + 1$
$x^{46} + 2x^{16} + 1$
$x^{46} + 2x^{40} + 1$
$x^{47} + 2x^{32} + 1$
$x^{53} + 2x^{40} + 1$
$x^{54} + 2x^{40} + 1$
$x^{55} + 2x^{32} + 1$
$x^{58} + 2x^{8} + 1$
$x^{62} + 2x^{40} + 1$
$x^{66} + 2x^{56} + 1$
$x^{67} + 2x^{56} + 1$
$x^{73} + 2x^{72} + 1$
$x^{74} + 2x^{24} + 1$
$x^{74} + 2x^{48} + 1$
$x^{77} + 2x^{16} + 1$
$x^{78} + 2x^{64} + 1$
$x^{81} + 2x^{40} + 1$
$x^{82} + 2x^{80} + 1$
$x^{83} + 2x^{32} + 1$
$x^{83} + 2x^{56} + 1$
$x^{85} + 2x^{16} + 1$
$x^{89} + 2x^{64} + 1$
$x^{90} + 2x^{56} + 1$
$x^{94} + 2x^{64} + 1$
$x^{95} + 2x^{48} + 1$
$x^{97} + 2x^{16} + 1$
$x^{99} + 2x^{80} + 1$
$x^{103} + 2x^{56} + 1$
$x^{106} + 2x^{80} + 1$
$x^{107} + 2x^{32} + 1$
$x^{107} + 2x^{96} + 1$
$x^{107} + 2x^{104} + 1$
$x^{109} + 2x^{88} + 1$
$x^{109} + 2x^{96} + 1$
$x^{110} + 2x^{64} + 1$
$x^{110} + 2x^{88} + 1$
$x^{115} + 2x^{32} + 1$
$x^{118} + 2x^{40} + 1$
$x^{121} + 2x^{40} + 1$
$x^{121} + 2x^{120} + 1$
$x^{127} + 2x^{8} + 1$
$x^{131} + 2x^{48} + 1$
$x^{131} + 2x^{104} + 1$
$x^{133} + 2x^{88} + 1$
$x^{134} + 2x^{112} + 1$
$x^{137} + 2x^{136} + 1$
$x^{138} + 2x^{104} + 1$
$x^{139} + 2x^{80} + 1$
$x^{141} + 2x^{64} + 1$
$x^{141} + 2x^{136} + 1$
$x^{142} + 2x^{40} + 1$
$x^{145} + 2x^{24} + 1$
$x^{145} + 2x^{64} + 1$
$x^{145} + 2x^{72} + 1$
$x^{146} + 2x^{144} + 1$
$x^{147} + 2x^{8} + 1$
$x^{147} + 2x^{104} + 1$
$x^{154} + 2x^{32} + 1$
$x^{154} + 2x^{104} + 1$
$x^{157} + 2x^{88} + 1$
$x^{159} + 2x^{32} + 1$
$x^{162} + 2x^{80} + 1$
$x^{163} + 2x^{80} + 1$
$x^{163} + 2x^{104} + 1$
$x^{165} + 2x^{88} + 1$
$x^{166} + 2x^{64} + 1$
$x^{166} + 2x^{88} + 1$
$x^{166} + 2x^{112} + 1$
$x^{167} + 2x^{96} + 1$
$x^{169} + 2x^{24} + 1$
$x^{169} + 2x^{64} + 1$
$x^{170} + 2x^{32} + 1$
$x^{178} + 2x^{128} + 1$
$x^{178} + 2x^{152} + 1$
$x^{179} + 2x^{104} + 1$
$x^{179} + 2x^{120} + 1$
$x^{181} + 2x^{40} + 1$
$x^{181} + 2x^{144} + 1$
$x^{185} + 2x^{64} + 1$
$x^{187} + 2x^{8} + 1$
$x^{187} + 2x^{32} + 1$
$x^{187} + 2x^{56} + 1$
$x^{190} + 2x^{96} + 1$
$x^{191} + 2x^{120} + 1$
$x^{193} + 2x^{64} + 1$
$x^{193} + 2x^{112} + 1$
$x^{194} + 2x^{24} + 1$
$x^{194} + 2x^{32} + 1$
$x^{195} + 2x^{104} + 1$
$x^{198} + 2x^{160} + 1$
$x^{201} + 2x^{88} + 1$
$x^{203} + 2x^{8} + 1$
$x^{203} + 2x^{32} + 1$
$x^{203} + 2x^{200} + 1$
$x^{205} + 2x^{144} + 1$
$x^{205} + 2x^{168} + 1$
$x^{206} + 2x^{112} + 1$
$x^{209} + 2x^{40} + 1$
$x^{209} + 2x^{160} + 1$
$x^{214} + 2x^{64} + 1$
$x^{214} + 2x^{192} + 1$
$x^{214} + 2x^{208} + 1$
$x^{218} + 2x^{176} + 1$
$x^{218} + 2x^{192} + 1$
$x^{218} + 2x^{200} + 1$
$x^{225} + 2x^{16} + 1$
$x^{227} + 2x^{120} + 1$
$x^{227} + 2x^{216} + 1$
$x^{229} + 2x^{72} + 1$
$x^{230} + 2x^{64} + 1$
$x^{234} + 2x^{104} + 1$
$x^{235} + 2x^{152} + 1$
$x^{238} + 2x^{232} + 1$
$x^{239} + 2x^{24} + 1$
$x^{239} + 2x^{56} + 1$
$x^{239} + 2x^{96} + 1$
$x^{239} + 2x^{104} + 1$
$x^{241} + 2x^{88} + 1$
$x^{242} + 2x^{80} + 1$
$x^{242} + 2x^{240} + 1$
$x^{250} + 2x^{104} + 1$
$x^{253} + 2x^{136} + 1$
$x^{253} + 2x^{184} + 1$
$x^{254} + 2x^{16} + 1$
$x^{16} + 2x^{8} + 2$
$x^{32} + 2x^{8} + 2$
$x^{32} + 2x^{16} + 2$
$x^{32} + 2x^{24} + 2$
$x^{48} + 2x^{8} + 2$
$x^{48} + 2x^{40} + 2$
$x^{64} + 2x^{16} + 2$
$x^{64} + 2x^{24} + 2$
$x^{64} + 2x^{32} + 2$
$x^{64} + 2x^{40} + 2$
$x^{64} + 2x^{48} + 2$
$x^{96} + 2x^{16} + 2$
$x^{96} + 2x^{80} + 2$
$x^{112} + 2x^{8} + 2$
$x^{112} + 2x^{104} + 2$
$x^{128} + 2x^{32} + 2$
$x^{128} + 2x^{48} + 2$
$x^{128} + 2x^{56} + 2$
$x^{128} + 2x^{64} + 2$
$x^{128} + 2x^{72} + 2$
$x^{128} + 2x^{80} + 2$
$x^{128} + 2x^{96} + 2$
$x^{144} + 2x^{56} + 2$
$x^{144} + 2x^{88} + 2$
$x^{160} + 2x^{40} + 2$
$x^{160} + 2x^{120} + 2$
$x^{176} + 2x^{40} + 2$
$x^{176} + 2x^{136} + 2$
$x^{192} + 2x^{32} + 2$
$x^{192} + 2x^{160} + 2$
$x^{208} + 2x^{56} + 2$
$x^{208} + 2x^{152} + 2$
$x^{224} + 2x^{16} + 2$
$x^{224} + 2x^{104} + 2$
$x^{224} + 2x^{120} + 2$
$x^{224} + 2x^{208} + 2$
$x^{240} + 2x^{8} + 2$
$x^{240} + 2x^{232} + 2$
$x^{256} + 2x^{40} + 2$
$x^{256} + 2x^{64} + 2$
$x^{256} + 2x^{96} + 2$
$x^{256} + 2x^{112} + 2$
$x^{256} + 2x^{128} + 2$
$x^{256} + 2x^{144} + 2$
$x^{256} + 2x^{160} + 2$
$x^{256} + 2x^{192} + 2$
$x^{256} + 2x^{216} + 2$

Experiments on the Linear Algebra Step in the Number Field Sieve

Kazumaro Aoki[1], Takeshi Shimoyama[2], and Hiroki Ueda[1,*]

[1] NTT, 3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
[2] Fujitsu Laboratories Ltd, 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki-shi, Kanagawa-ken, 211-8588 Japan

Abstract. This paper shows experimental results of the linear algebra step in the number field sieve on parallel environment with implementation techniques. We developed an efficient algorithm that shares the sum of vectors in each node, and the network structure among the nodes only requires to include a ring. We also investigated the construction of a network for the linear algebra step. The construction can be realized through switches and network interface cards, whose prices are not expensive. Moreover, we investigated the implementation of the linear algebra step using various parameters. The implementation described in this paper was used for the integer factoring of a 176 digit number by GNFS and a 274 digit number by SNFS.

Keywords: integer factoring, number field sieve, block Lanczos, parallel computation, ethernet.

1 Introduction

This paper presents the results of our implementation of the linear algebra step for the number field sieve [1], which is the asymptotically-fastest integer factoring algorithm known. The linear algebra step in the number field sieve seems to require a shorter execution time than the sieving step based on past experiments, but the linear algebra step also theoretically dominates the execution time of the number field sieve. Reviewing the recent factoring records of the general number field sieve [2], the linear algebra step requires approximately one fourth of the total complexity, and some articles, for example [3], estimate that the linear algebra step will dominate the execution time of factoring larger numbers. When comparing the linear algebra step to the sieving step one will find that the difficulty lies in the parallel computation. A simple split of the computation seems to require a high-bandwidth and low latency network connection. However, it seems that the problem can gradually be solved because recent factoring reports show that a cluster of PCs can be used, for example [4], but the details were not given.

This paper shows our implementation in a parallel environment, and describes the extensive experiments based on the implementation of the linear algebra

[*] He is currently working for Nippon Telegraph and Telephone West Corporation.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 58–73, 2007.
© Springer-Verlag Berlin Heidelberg 2007

step. The experiments include the tradeoff between sieving and linear algebra steps using block Lanczos algorithm, timing details in the linear algebra steps for factoring 176 digit number.

2 Preliminaries

This section defines the notations used in the paper through the explanation of algorithms in the number field sieve. We sometimes omit the correct and precise description whose property is not used for the following sections.

2.1 Number Field Sieve

This section briefly describes the number field sieve that is relevant to the scope of this paper. Details regarding the algorithm can be found in, e.g. [1].

Let N be a composite number and it will be factored. Find irreducible polynomial $f(X) \in \mathbb{Z}[X]$ and its root r such that $f(r) \equiv 0 \pmod{N}$. Let $F_R = \{p \mid p\text{: prime},\ p \leq B_R\}$ and $F_A = \{(q,s) \mid q\text{: prime},\ q \leq B_A,\ f(s) \equiv 0 \pmod{q}\}$. We call $F = F_R \cup F_A$ *factor base*. The purpose of the sieving step is to collect coprime pairs $(a,b) \in \mathbb{Z}^2$ such that $N_R(a,b) = |a + br| = \prod_{p \in F_R} p^{e_p}$ and $N_A(a,b) = |(-b)^{\deg f} f(-a/b)| = \prod_{(q,s) \in F_A} q^{e_q}$. Such a coprime pair, (a,b), is called a *relation*. The number of collected relations should roughly exceed $\#F$, where $\#$ denotes the number of elements in the set.

For each relation (a,b), let $v_{(a,b)}$ be a vector that consists of exponents $\bmod 2$ in the factorization of $N_R(a,b)$ and $N_A(a,b)$, and V be the matrix that consists of such vectors $v_{(a,b)}$. The purpose of the linear algebra step is to find several non-trivial solutions x that satisfy $Vx = 0$. The matrix produced by the world-record class factorization is sparse but gigantic. The linear algebra step usually consists of two steps:

1. Filtering that generates a sparse matrix M from V[1].
2. Solving the sparse matrix M that is generated by Step 1.

Mathematically speaking, the filtering step is a kind of Gaussian elimination. Before using the algorithm to solve a sparse matrix, this kind of clever Gaussian elimination greatly reduces the execution time for solving the matrix. We can use the block Lanczos [6] or block Wiedemann [7] algorithms to solve the sparse matrix generated by the filtering step. We do not know which algorithm is superior, but block Lanczos was used for recent factorization records[2] except for the RSA-200 factorization [2].

[1] According to [5], the size of V is 455 989 949×328 707 916 and that of M is 8 526 749×8 525 751 and its weight is 1 707 545 745, for example.

[2] http://www.crypto-world.com/FactorWorld.html

2.2 Block Lanczos Algorithm

The block Lanczos algorithm solves linear equation $AX = B$ of symmetric matrix A over GF(2) and nonzero vector B. This algorithm is more suitable for a sparse matrix than a general purpose algorithm. Since the algorithm can find several Xs simultaneously for each B, we regard an element in B as a small vector over GF(2). We call the element *word block*.

The block Lanczos algorithm requires a symmetric matrix A and a nonzero vector B as an input; however, output matrix M of the filtering step is probably not symmetric, and a solution must be obtained in the case of $B = 0$. Then, in order to use the block Lanczos algorithm, we set the input such that $A = M^T M$ where M^T is the transpose matrix of M and $B = AC$ for any nonzero random vector C. In this case, if X is an output of the block Lanczos algorithm, we will obtain the final non-trivial solution $X - C$.

Algorithm 1 (Block Lanczos algorithm)
Input: $m \times m$ matrix A and nonzero vector B of t-bit word blocks
Output: Vector X such that $\dim(AX - B) \ll t$

1. Set up the initial values. Let $V_1 \leftarrow B$, $U_0 \leftarrow 0$, and $W_0 \leftarrow 0$.
2. Loop the following procedure from $i = 1$.
   (a) Extract maximum subvector W_i from V_i such that $W_i \supset U_{i-1}$ and $\det((W_i, W_i)_A) \neq 0$, where $(W_1, W_2)_A = W_1^T A W_2$.
   (b) If $W_i = \emptyset$ then exit from the loop.
   (c) Let $U_i \leftarrow V_i \setminus W_i$.
   (d) Set $C_{i,j}$ for $(W_j, W_j)_A{}^{-1}(W_j, [U_i|AW_i])_A$ and calculate the following vector, where $[U_i|AW_i]$ is the concatenation of vectors U_i and AW_i.

$$V_{i+1} \leftarrow [U_i|AW_i] + W_i C_{i,i} + W_{i-1} C_{i,i-1} + W_{i-2} C_{i,i-2}$$

3. Let $m = i$ and output vector X calculated as follows.

$$X \leftarrow \sum_{j=0}^{m-1} W_j (W_j, W_j)_A{}^{-1} (W_j, B)$$

We note the complexity of the block Lanczos algorithm for a single processor as follows. For an $m \times l$ matrix with d nonzero elements in each columns for average and a vector with a t-bit block length, $O(mld) + O(mlt)$ bit memory accesses and $O(mld) + O(mlt^2)$ bitwise logical operations are required. For the details on the block Lanczos algorithm, refer to [6].

2.3 Ethernet and Its Switch

This section describes the characteristics of the ethernet and its switch relevant of this paper.

The ethernet is the most typical way to connect between PCs. At the time of writing the paper, one ethernet port of a PC can bidirectionally transmit data and its throughputs are 1 Gb/s per one-way, and it is available at cheap price. The most simple ethernet network is realized by a switch connected through cables with PC. It looks like star architecture, but the ethernet is logically bus architecture. A switch can exchange data at the same time among all PCs connected to the switch. So, the network can be considered as complete graph among all connected PCs. We explain the detail using the following example. Assume that PC A, B, and C are connected to the switch. A can send data to B, and B can send data to C, and C can send data to A, at the same time, and the throughput of all connections are probably 1 Gb/s, but this depends on the specification of the switch.

The following limitations should be concerned when constructing a network using ethernet. If more PCs are required to be connected to an ethernet than the number of ports of a switch, switches with cross cables can be used to extend the number of ports in the ethernet. In this construction, only 1-to-1 data transmission is available through plural switches at a moment. Moreover, a loop in one ethernet is forbidden because of bus architecture.

A network interface controller in a PC has usually sufficient amount of buffer for transmitting and receiving data. That is, the network transactions do not require much CPU cycles. A part of the network communication time can be hidden in the computation time, using thread programming.

3 Sharing of the Sum of Vectors

For implementing linear algebra step, an efficient algorithm to share the sum of vectors that is stored in each node is useful. We propose an algorithm to accomplish this.

Let l-dimensional vector $v^{(i)} = (v_0^{(i)}, v_1^{(i)}, \ldots, v_{l-1}^{(i)}) \in S^l$, and S be a commutative semi-group S. For the scope of this paper, S is GF(2). So, we use the notation $\bigoplus$ instead of $\sum$. Assume that node i holds $v^{(i)}$ and $l \gg n$, where n is the number of nodes. The goal of the following algorithm is to share $\bigoplus_{i=0}^{n-1} v^{(i)}$.

The simplest way to achieve the goal is to employ the following.

1. For $i \leftarrow 1, 2, \ldots, n-1$, node i sends $v^{(i)}$ to node 0.
2. Node 0 computes $s \leftarrow \bigoplus_{i=0}^{n-1} v^{(i)}$, and sends s to all nodes.

If we assume that one element of S can be sent in time 1, the above algorithm requires time $2l(n-1)$. Of course, there are more sophisticated algorithms. The most famous algorithm is executed based on the tree structure. The algorithm requires time $2l \lceil \log_2 n \rceil$ and the network should include many edges.

The above algorithms compute the sum in a node, but the sum may be computed at any node. We develop the following algorithm. Let $(b_0^{(i)}, b_1^{(i)}, \ldots, b_{n-1}^{(i)}) = v^{(i)}$, and each $b_j^{(i)}$ contains roughly the same number of elements ($\approx \frac{l}{n}$).

Algorithm 2 (Vector Sum Sharing)

Before: Node i $(0 \le i < n)$ has the vector $(b_0^{(i)}, b_1^{(i)}, \ldots, b_{n-1}^{(i)})$ $(= v^{(i)})$.

After: All nodes have $\bigoplus_{i=0}^{n-1} v^{(i)}$.

```
for $\tau \leftarrow 0$ to $2n-3$
    for $i \leftarrow 0$ to $n-1$ do parallel
        node $i$ sends $b_{\tau+i}^{(i)}$ to node $i-1$
        node $i-1$
            if $\tau \le n-2$   /* generating sum */
                receive $b_{\tau+i}^{(i)}$ from node $i$ and $b_{\tau+i}^{(i-1)} \leftarrow b_{\tau+i}^{(i-1)} + b_{\tau+i}^{(i)}$
            else   /* $\tau \ge n-1$. distributing sum */
                receive $b_{\tau+i}^{(i)}$ from node $i$ and $b_{\tau+i}^{(i-1)} \leftarrow b_{\tau+i}^{(i)}$
```

mod n should be applied to the node number and the suffix of $b^{(i)}$. The proposed algorithm requires time $2(n-1)\left\lceil \frac{l}{n} \right\rceil$, and the network structure should include a ring[3] with length n and full-duplex.

4 Block Lanczos Algorithm for a Parallel Environment

The block Lanczos algorithm, described in Algorithm 1, essentially uses three kinds of operations: The first is the multiplication of a large sparse matrix and a vector, and the other two are the vector operations, i.e., an inner product of two vectors and a product of a vector and a tiny matrix. To implement the block Lanczos algorithm on a PC cluster, we must consider both parallelization of the above procedures and network connectivity. In this section, we discuss the implementation of these operations. Most of the algorithm are not new, but we describe the details, because what is implemented is important to understand the timing data described in Section 6.

4.1 Multiplication of a Large Sparse Matrix and a Vector

The multiplication of a large sparse matrix and a vector is one of the most time consuming parts in the block Lanczos algorithm. We use the following notations. An $m \times l$ matrix M on GF(2) has ld number of nonzero elements, (so called "weight" of the matrix), so it has d nonzero elements in each column on average and we assume $d \ll l$. Each element in l-dimensional vector X is a word with t-bit data (so called "length" of the vector). In this section, we consider the calculation of $Y = M^T(MX)$ in parallel.

[3] Initially, we believed that a complete graph is required to execute the algorithm, however, Dr. Yamamoto of NTT pointed out that a ring network is sufficient.

Data format of the matrix. Since we consider a sparse matrix based on GF(2), the matrix should be represented by the index of the coordinates of nonzero elements. Moreover, we can assume that each row and column in the matrix has at least one element. From the implementation viewpoint, the property is not required, but the property is important to get the enough number of effective solutions. Our data format of the matrix can be described as the repetition of the format $(m, a_1, \ldots, a_m)$, whose m is the number of nonzero elements of each column and $a_1, \ldots, a_m$ are the coordinates of the row of each of nonzero element in the columns. Since data access of the matrix is sequential, by counting the number of repetition of the above format, an index of a column can easily be recovered. Then the index of the coordinates of the columns can be omitted in the matrix data. Using this data format, $l(1+d)$ words are required.

4.2 Matrix Operation in a Parallel Environment

This section describes our implementation for matrix operations, which would help to understand Section 6. Note that the most of the ideas comes from [8].

We simulate the operation for the computation of $Z = MX$. Let Z be a zero vector at first. For each element a_i in the j-th format of the matrix data, the j-th coordinate element X_j in vector X is added to the a_i-th coordinate element Z_{a_i} in vector Z. This operation requires one sequential memory access (for X_j) and one random memory access (for Z_{a_i}). To compute $Y = M^T Z$, Z_{a_i} is added to Y_j, where element a_i is in the j-th format of the matrix data. This operation requires one random memory access (for Z_{a_i}) and one sequential memory access (for Y_j). By taking into account two sequential memory accesses of the matrix data, four sequential memory accesses and two random memory accesses are required according to each element in the matrix data format.

Now, we consider parallel computation of the matrix operation. For simplicity, we assume that the nonzero elements in the matrix are uniformly distributed, and the number of nodes is square ($n \times n$).

Comment 1 . Actually, nonzero elements in the input matrix data may not distribute uniformly. Partitioning matrices that have almost the same weights can be constructed through the systematic shuffling of the matrix columns and rows.

A large matrix must be partitioned into each node. To accomplish this, the following methods are considered.

1. Partition the matrix into rectangles based on the matrix row (or column).
2. Partition the matrix into squares based on the matrix row and column.

We need to consider not only the parallel operation of the multiplication of the matrix and the vector, but also the vector sum sharing using the network communications. Table 1 shows for the available number of nodes, the number of operations for the random memory accesses, the sequential memory accesses, and the amount of data in the network transaction in a parallel environment.

We assume that Algorithm 2 is used for the network transactions. Both partitioning methods have the same number of memory access operations[4]. However, the rectangle partitioning method requires a higher level of complexity for the network transaction part compared to that for the square partitioning method. Therefore, we select the square partitioning method in the implementation.

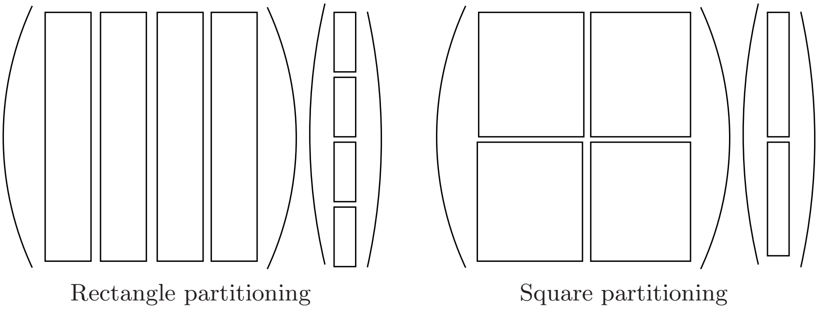


Fig. 1. Partitioning of large matrix

Table 1. Bitwise Complexity for $M^T MX$ on Each Node

Partitioning	Rectangle	Square
number of random memory accesses for a vector	$2dlt/n^2$	$2dlt/n^2$
number of sequential memory accesses for a vector	$2dlt/n^2$	$2dlt/n^2$
number of sequential memory accesses for a matrix	$2(1+d)wl/n^2$	$2(n+d)wl/n^2$
number of network transaction	$2l(n^2-1)t/n^2$	$4l(n-1)t/n^2$

n^2: Number of nodes l: Dimension of a vector
t: Bit length of a vector (ex. 256)
d: Average of nonzero elements of each column in a matrix
w: Bit length of the word in the matrix format (ex. 32)

In Algorithm 3, we show the procedure for the multiplication of a large sparse matrix and a vector in a parallel environment using the square partitioning method. Let $U_{i,j}$ $(i, j = 0, \ldots, n-1)$ be $\lceil m/n \rceil \times \lceil l/n \rceil$ submatrices from the matrix M divided based on square partitioning. Each (i, j)-th node holds the submatrix $U_{i,j}$ in memory[5]. For vector X, let $X_0, \ldots, X_{n-1}$ be the vectors obtained by partitioning l-dimensional vector X into vectors with $\lceil l/n \rceil$ elements. Let $X_{i,j}$ be the vectors obtained by partitioning vector X_i into vectors with

[4] Using our data format for storing the matrix, for generating the sub-matrices by square partitioning into $n \times n$, we need to set the data "m" of the number of nonzero elements in each columns of the sub-matrices. Therefore, it is a few difference between the number of the memory access of the square partitioning and that of the rectangle partitioning.

[5] Using square partitioning, there is some possibility that a column with null elements appears in $U_{i,j}$; therefore, careful coding must be performed in order to deal with these submatrices.

$\lceil \lceil l/n \rceil / n \rceil$ elements. We use zero-padding for the shortage of the vector elements. Let vector $X_{i,j}$ be stored in the (i,j)-th node. Output vector Y is also partitioned into $Y_{i,j}$ and stored in the (i,j)-th node in the same manner.

Algorithm 3 (Multiplication of a Matrix and a Vector)

```
1. For each $i = 0, \dots, n-1$, vector $X_i$, which is a concatenation of the vectors
   from $X_{i,0}$ to $X_{i,n-1}$, will be stored in $n$ nodes from $(i,0)$ to $(i,n-1)$.
2. In all the nodes, execute the multiplication of submatrix $U_{i,j}$ and vector $X_i$,
   and store the results in $\lceil m/n \rceil$-dimensional vector $Z_{i,j} \leftarrow U_{i,j} X_i$.
3. For each $j = 0, \dots, n-1$, share the sum of vectors $Z_j \leftarrow \bigoplus_{i=0}^{n-1} Z_{i,j}$ with
   $n$-nodes from $(0,j)$ to $(n-1,j)$.
4. In all the nodes, execute the multiplication of submatrix $U_{i,j}^T$ and vector $Z_j$,
   and store the results in $\lceil l/n \rceil$-dimensional vector $W_{i,j} \leftarrow U_{i,j}^T Z_j$.
5. For each $i, j = 0, \dots, n-1$, store $\lceil \lceil l/n \rceil / n \rceil$-dimensional vector $Y_{i,j}$ on the
   $(i,j)$-th node which is a part of the sum of vectors $Y_j \leftarrow \bigoplus_{j=0}^{n-1} W_{i,j}$.
```

Note that the resulting vectors are stored in the suitable nodes for computing $Y \leftarrow M^T Z$ after executing Algorithm 3 for $Z \leftarrow MX$, that is, we do not need to store M^T.

4.3 Vector Operations

This section shows an implementation of the vector operations that appear in the block Lanczos algorithm, which would help to understand Section 6.

We set each vector as an l-dimensional vector with coordinates of t-bit words (ex. $t = 256$). In the block Lanczos algorithm, there are two kinds of vector operations, one is an inner product, and the other is a product of a vector and a *tiny-matrix* with $t \times t$ elements. The output of the inner product of two vectors is a tiny-matrix, and the output of the product of a vector and a tiny-matrix is a vector. In this implementation, both vector operations are achieved using the combination of an XOR operation and a 1-bit rotation on t-bit words. Let $work[j]$ be the j-th row of a temporal tiny-matrix of $work$, and $a^{\lll j}$ denotes the j-bit cyclic left rotation of word a.

Algorithm 4 (Inner Product)
Input: l-dimensional vectors X and Y, Output: $t \times t$ tiny-matrix $mat = X^T Y$

```
1. For temporary tiny-matrix $work$, initialize the elements in $work$ as zero.
2. For $i \leftarrow 0, \dots, l-1$, execute the following.
   (a) Set $a \leftarrow X[i]$ and $b \leftarrow Y[i]$.
   (b) For $j \leftarrow 0, \dots, t-1$, $work[j] \leftarrow work[j] \oplus ((a^{\lll j}) \& b)$.
3. For each $i, j \leftarrow 0, \dots, t-1$, $mat[(i+j) \bmod t][j] \leftarrow work[i][j]$.
```

Algorithm 5 (Product of a Vector and a Tiny-matrix)
Input: l-dimensional vector X and a tiny-matrix mat
Output: l-dimensional vector $Y = X \cdot mat$

```
1. For $i, j \leftarrow 0, \ldots, t-1$
   $work[i][j] \leftarrow mat[(i+j) \bmod t][j]$.
2. Initialize all the elements in the vector $Y$ as zero.
3. For $i \leftarrow 0, \ldots, l-1$, execute the following.
   (a) Set $a \leftarrow X[i]$.
   (b) For $j \leftarrow 0, \ldots, t-1$, $Y[i] \leftarrow Y[i] \oplus ((a^{\lll j}) \& work[j])$.
```

On Algorithms 4 and 5, if t is small and the program can use large cache memory, the table lookup method described in [7] may improves the performance.

Vector Operations in a Parallel Environment. Let X and Y be l-dimensional vectors. We assume that vectors X and Y are partitioned into $\lceil \lceil l/n \rceil / n \rceil$-dimensional vectors $X_{i,j}$ and $Y_{i,j}$, which are stored in the corresponding nodes. The inner product of vectors X and Y can be executed in the following manner.

Algorithm 6 (Inner Product in a Parallel Environment)

```
1. Calculate the inner product of $X_{i,j}$ and $Y_{i,j}$ on all nodes.
2. Share the sum of the inner products of all nodes.
```

Since the results of the inner products on all nodes are tiny-matrices, which are very very tiny, the network transaction time for the inner products using parallel computation can be ignored. The product of a vector and a tiny matrix can be executed in a parallel environment in almost the same manner as the procedure for the inner product.

4.4 Network Construction

According to the analysis in Section 4.1, a squared number of nodes is appropriate to execute the block Lanczos algorithm, and each column and row should be connected to a ring network. Figure 2 shows an example of 4 × 4 nodes. If the number of nodes is less than 16, a switch connected with all nodes is a good candidate for use in the construction of the network, because we can consider that the network is complete. If the number of nodes is large, for example, 64, a switch is not a good candidate for use in the construction of the network, because the price of a switch with 64 ports is expensive.

For the case that the number of nodes is large, we prepared several switches and the nodes have two network interface cards (NICs). We assign a switch to each row or column, i.e., the network for $n \times n$ nodes is constructed using a $2n$-many switches, each having n ports, and each node has two NICs. In general, if the total price of $2n$-many switches, each having n ports, and $2n^2$ NICs is less than the price of an n^2-port switch and n^2 NICs, our construction is less expensive. Figures 3 and 4 are examples of a network for 8 × 8 nodes, using the idea with realistic configuration parameters. Each square consists of a node, and

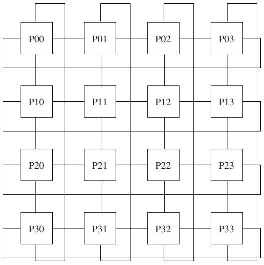


Fig. 2. Network connection for 4 × 4 nodes

1-	1-	1-	1-	12	12	12	12
1-	1-	1-	1-	12	12	12	12
1-	1-	1-	1-	12	12	12	12
1-	1-	1-	1-	12	12	12	12
12	12	12	12	-2	-2	-2	-2
12	12	12	12	-2	-2	-2	-2
12	12	12	12	-2	-2	-2	-2
12	12	12	12	-2	-2	-2	-2

Fig. 3. 8 × 8 nodes with two 48-port switches

1-	12	13	14	15	16	17	18
12	2-	23	24	25	26	27	28
13	23	3-	34	35	36	37	38
14	24	34	4-	45	46	47	48
15	25	35	45	5-	56	57	58
16	26	36	46	56	6-	67	68
17	27	37	47	57	67	7-	78
18	28	38	48	58	68	78	8-

Fig. 4. 8 × 8 nodes with eight 15-port switches

a digit means that the node is connected to a specific switch#. The construction of the network in Fig. 3 requires 96 NICs and 2-many switches, each having 48 ports, and Fig. 4 construction requires 120 NICs and 8-many switches, each having 15 ports, while the simplest construction requires 64 NICs and one switch with 64 ports.

Comment 2. We can construct a network that is appropriate to execute the block Lanczos algorithm without a switch using four NICs at each node and cross-cables. However, currently it seems to cost expensive price to use this construction to compose any large network such as one comprising hundreds or thousands of nodes. The strategy of using four NICs with a cross-cable, however, may be applicable to a portion of the network construction.

5 Other Techniques

This section describes other optimization techniques that do not directly relate to a parallel environment.

5.1 Inner Product Using 128-Bit XMM Registers

As described in Section 2.2, the inner product incurs one of the heaviest computation loads in the block Lanczos algorithm. Focusing on the inner product subroutine, generating all rotation patterns $(x, x^{\lll 1}, \ldots, x^{\lll t-1})$ dominates the performance, when we use Algorithms 4 and 5, where $\lll$ denotes the left rotation and t is the word length used in the block Lanczos algorithm. We found that the code in Fig. 5 efficiently computes 1-bit rotation of the XMM register pairs on a Pentium 4 [Northwood] in the case of $t = 128$. Note that all rotation patterns are generated using a $t/2-1$ calls of the code. Using the idea in Fig. 5, we developed an efficient rotation code for $t = 256$.

5.2 Fault Tolerance by Orthogonality Check

In our consideration of the block Lanczos algorithm, we assumed that the PC clusters function without any computational error. When operating the

```
; compute: xmm1 <<<= 1 and xmm5 <<<= 1
; assumption1: xmm5 == xmm1 <<< 64
; assumption2: MSB of xmm2 and xmm6 is cleared
por      xmm2,    xmm1    ; MSB(xmm2) = MSB(xmm1)
psrlq    xmm2,    63
por      xmm6,    xmm5    ; MSB(xmm6) = MSB(xmm5)
psrlq    xmm6,    63
paddq    xmm1,    xmm1    ; xmm1 <<= 1
paddq    xmm5,    xmm5    ; xmm5 <<= 1
paddq    xmm1,    xmm6    ; fill LSB with crossing
por      xmm5,    xmm2
```

Fig. 5. Rotation of XMM register pair on Pentium 4

computer systems for several weeks, however, some errors may occur such as memory error, network transmission error, and CPU error. Refer Section 6.2 as an example.

If an error occurs, continuing the computation becomes meaningless. Therefore, we must prepare some mechanism such as fault tolerance, i.e., fault detection, and recover the data from the previous computations. Based on our experience, we consider that the following algorithm is sufficient to detect errors and recover the computation data.

Algorithm 7 (Orthogonality Check)

1. Periodically, backup inner variables to a file.
2. Check the following condition just after Step 2a in Algorithm 1.

$$(B, W_i)_A = 0 \tag{1}$$

3. If the condition does not hold, load the backup file and restart from that step.

The cost for Step 2 in the above algorithm is equivalent to that of one inner product of vector W and vector AW, which is less than 5% of total execution time of block Lanczos algorithm according to our experience. If the orthogonality is broken at some time, Eq. (1) never holds for the following computation. We do not need to confirm Eq. (1) for every loop. Note that we stored inner variables every 500 loops and confirmed Eq. (1) every loop for the experiments described in Section 6.

6 Experiments on c176

According to the previous sections, we implemented a block Lanczos algorithm. C language with XMM intrinsic functions is used except for inner product, and LAM/MPI[6] is used for the communication.

[6] http://www.lam-mpi.org/

We used data obtained in the c176 factorization [5]. c176 is the 176-digit composite cofactor of $11^{281} + 1$ in the Cunningham table[7]. We selected the block length of 256 in the block Lanczos algorithm. Note that we removed 224 heavy rows, which are probably associated to small factor bases, before executing the block Lanczos implementation. These rows do not include the rows for quadratic characters. These rows can be recovered through Gaussian elimination after finding sufficiently many solutions. Refer to [5] for detailed parameters used in the factorization of c176.

6.1 Our Cluster

We constructed the cluster for factoring integers organized as described hereafter. We used a maximum of 36 nodes, where each node consists of 1 CPU with 2GB RAM using the FreeBSD 4R operating system. We used 32 Pentium 4 3.2GHz [Northwood] PCs, 2 Pentium 4 Xeon 2.8GHz [Northwood] PCs, and 2 Pentium 4 Xeon 3.6GHz [Prescott] PCs employing 32-bit processing. Hyperthreading technology is enabled on all CPUs, but we only used one thread at a node. These nodes are connected through gigabit Ethernet, and the network is structured based on the concept in Section 4.4. We used 3-many switches, each having 24 ports, to construct the network. Each node has 1 on-board and 1 PCI network interface. The PCI bus length is 32-bit and functions at 33MHz. Note that, at the time of the cluster construction, dual CPU and dual core were not widely available or not cheap, and we already had the block Lanczos code that is optimized for Pentium 4 [Northwood]. We did not choose AMD CPU as main components of the cluster.

We confirmed the performance of gigabit Ethernet using `ttcp`[8]. The results are described in Table 2. Because the ideal throughput is only 32-bit × 33 MHz = 1056 Mb/s, we cannot occupy all the network resources. Moreover, we confirmed that the switches used in the cluster can independently send data to each node. The maximum throughput described in Table 2 can be achieved even in the case of full-duplex communication for 12 nodes.

Table 2. Gigabit Ethernet Throughput

Duplex	Source	Destination	Speed (Mb/s)
Half	On-board	On-board	884
Half	PCI	PCI	602
Half	On-board	PCI	750
Half	PCI	On-board	602
Full	On-board	On-board	566
Full	PCI	PCI	318
Full	PCI(On-board)	On-board(PCI)	556(862)

[7] `http://homes.cerias.purdue.edu/~ssw/cun/`

[8] `http://ftp.arl.mil/~mike/ttcp.html`

6.2 Tradeoff Between Sieve and Block Lanczos

When actually factoring c176, we used lattice sieving [9] as a sieving method[9]. 10.6M special-Qs from the range of $30.5\text{M} \leq Q \leq 226.0\text{M}$, where M represents 10^6, were used. When factoring a 248-digit composite using SNFS [10], many hardware troubles were experienced, i.e., bit-flipping during the block Lanczos computation, and the block Lanczos algorithm was needed to be executed several times[10]. At that time, our block Lanczos code did not have error detection mechanism such as Algorithm 7. Considering this trouble, we wanted to reduce the matrix, which is the input for the block Lanczos algorithm, to as small as possible. To accomplish this, a large number of relations was collected. As a result, the block Lanczos solution was obtained in the first execution.

From the viewpoint of minimizing the factoring time, it is worth considering the number of relations that should be collected to factor c176. We varied the number of special-Qs and estimated the execution time. Table 3 summarizes these varied sizes. The "Number of sp-Q" column represents the number of the small special-Qs that are used to construct the matrix. We cannot construct a matrix in which the number of rows is less than the number of columns in the case of 5.0M special-Qs.

Table 3. Number of Special-Q and Matrix Size

Number of			Block Lanczos input		
sp-Q	Yields	Unique relations	Rows	Columns	Weight
5.0M	266 055 527	226 579 930	14 422 588	13 534 462	—
5.5M	294 129 265	248 486 770	12 147 652	12 148 903	1 999 094 097
6.0M	322 054 398	269 959 762	11 069 972	11 071 197	1 757 293 033
7.0M	378 318 016	311 189 917	10 110 782	10 112 015	1 545 012 150
8.0M	434 328 082	352 916 978	9 499 644	9 500 869	1 439 900 177
10.6M	576 372 161	455 989 949	8 525 527	8 526 749	1 394 050 132

When sieving, various PCs were used. We completely occupied tens of PCs for this sieving, and we used idle time of other PCs. Pentium 4 3.2GHz [Northwood] chips are the most frequently used in these PCs. Their sieved resource can be scaled to 131 Pentium 4 3.2GHz [Northwood] PCs. Table 4 summarizes the number of days required for sieving and executing the block Lanczos algorithm assuming that 131 Pentium 4 3.2GHz [Northwood] PCs are used for sieving and our cluster is used for the block Lanczos algorithm. According to the table, 6.0M special-Qs optimize the total execution time for our computational resources. Note that we did not fully execute the block Lanczos algorithm for the matrices generated from a smaller number of special-Qs. We estimated the total execution

[9] Lattice sieving only sieves relation (a, b) whose $N_A(a, b)$ is divisible by Q. We call such Q *special-Q*.

[10] We bought very cheap PCs. So, the components in a PC are not sufficiently examined at shop, and the shop hardly recognize this kind of errors, that will not detect in short time, say a week.

Table 4. Tradeoff Between Sieve and Block Lanczos

Number of sp-Q	Sieve days	Block Lanczos days	Total days
5.5M	15.6	10.4	26.0
6.0M	17.2	8.5	25.7
7.0M	20.4	7.0	27.4
8.0M	23.7	6.2	29.9
10.6M	27.0	5.3	32.3

time using the first 500 main iterations of our block Lanczos implementation. Sieve days for 5.5M-8.0M in Table 4 are estimation by interpolating the result of several sieving samples, so we may not choose the best range of special-Qs.

6.3 Details in Computation Time

We investigate the details in our block Lanczos implementation and the relationship between the number of nodes and the execution time using the matrix generated from 10.6M special-Qs. The results are summarized in Table 5. The number of execution days is estimated based on the first 100 main iterations of the block Lanczos implementation except for 6×6 nodes. We only used Pentium 4 3.2GHz [Northwood] PCs for 5×5, 4×4, and 3×3 nodes. We cannot execute our program on less than 3×3 nodes, because the required amount of RAM in each node exceeds 2 GB. We obtained "occupied RAM @node" from the output of `ps` command at the main iteration. We used `gcc` 3.4 for 6×6 nodes, and `icc` 8.1 is used for the other number of nodes tested. For 6×6 nodes, there is no record of the precise amount of occupied RAM, and we only have the results using the `gcc` compiler. We want to collect comparable numbers using a smaller number of nodes with the `icc` compiler, however a few of the nodes always seem to fail. Although repairs were made, other nodes failed subsequently. Thus, we, unfortunately, were unable to obtain data generated by 6×6 nodes.

Table 5. Ratio of Execution Time and Tradeoff Between Number of Nodes and Execution Time

Number of nodes	6×6	5×5	4×4	3×3
Matrix multiplication	33.0%	43.0%	47.6%	54.1%
Communication	21.1%	28.9%	24.4%	19.0%
Inner product	20.0%	22.3%	24.6%	26.6%
Synchronization	25.9%	0.7%	0.3%	0.3%
Occupied RAM @node	≈400 MB	470 MB	700 MB	1180 MB
Estimated number of days for execution	5.3	5.3	7.5	12.1

Note that used PCs for 6×6 nodes are not homogeneous and the used compiler for 6×6 nodes is different from other number of nodes.

If we focus on the synchronization row in Table 5, we need a long time for 6×6 nodes. This is because 6×6 nodes include various CPUs and our implementation does not consider the difference in each node.

Focusing on the matrix multiplication and inner product rows for column 5×5, 4×4, and 3×3 in Table 5, we summed up the total number of days for computation among all nodes, and the results are shown in the first and second rows in Table 6. According to the rows, we can confirm that the total computational complexity is almost the same among the different numbers of nodes.

Table 6. Invariant Numbers in Our Block Lanczos Implementation

Number of nodes $(n \times n)$	5×5	4×4	3×3
Number of days for $n^2 \times$ matrix multiplication	57.0	57.1	58.9
Number of days for $n^2 \times$ inner product	29.5	29.5	29.0
Number of days for $\dfrac{n^2}{n-1} \times$ communication	9.6	9.8	10.3

Focusing on the communication rows for columns 5×5, 4×4, and 3×3 in Table 5, we anticipate that the communication time is expected to be roughly proportional to the required time of Algorithm 2. Thus, the communication time is roughly proportional to $\dfrac{n-1}{n^2}$, when we have n^2 nodes. We show the communication time multiplied by $\dfrac{n^2}{n-1}$ in the third row in Table 6. These numbers seem roughly the same.

7 Concluding Remarks

We developed an efficient vector sum sharing algorithm, investigated the network construction, and implemented the block Lanczos algorithm in a parallel environment. Moreover, we extensively investigated the implementation using various parameters. Our experiments were done in 2005. We certainly notice that some of our experiments are incomplete. Moreover, we are aware that block Wiedemann algorithm may have some advantage than block Lanczos algorithm that we used for the linear algebra step. To fulfill the experimental data requires long time and it is hard to reconstruct the same computational environment as was at 2005. However, we believe that the results in this paper are useful for future factoring efforts.

References

1. Lenstra, A.K., Lenstra Jr., H.W. (eds.): The development of the number field sieve. Lecture Notes in Mathematics, vol. 1554. Springer, Heidelberg (1993)
2. Kleinjung, T.: rsa200 (2005), http://www.crypto-world.com/announcements/rsa200.txt

3. Silverman, R.D.: A cost-based security analysis of symmetric and asymmetric key lengths. RSA Laboratories' Bulletin, (revised 2001) (2000), http://www.rsasecurity.com/rsalabs/node.asp?id=2088
4. Leyland, P., Lenstra, A., Dodson, B., Muffett, A., Wagstaff, S.: MPQS with three large primes. In: Fieker, C., Kohel, D.R. (eds.) ANTS-V. LNCS, vol. 2369, pp. 446–460. Springer, Heidelberg (2002)
5. Aoki, K., Kida, Y., Shimoyama, T., Ueda, H.: GNFS176 (2005), http://www.crypto-world.com/announcements/c176.txt
6. Montgomery, P.L.: A block Lanczos algorithm for finding dependencies over GF(2). In: Guillou, L.C., Quisquater, J.J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 106–120. Springer, Heidelberg (1995)
7. Coppersmith, D.: Solving homogeneous linear equations over $GF(2)$ via block Wiedemann algorithm. Mathematics of Computation 62, 333–350 (2004)
8. Montgomery, P.L.: Distributed linear algebra. In: presentation slides for ECC2000 (2000) http://www.cacr.math.uwaterloo.ca/conferences/2000/ecc2000/montgomery.ps
9. Golliver, R.A., Lenstra, A.K., McCurley, K.S.: Lattice sieving and trial division. In: Huang, M-D.A., Adleman, L.M. (eds.) ANTS-I. LNCS, vol. 877, pp. 18–27. Springer, Heidelberg (1994)
10. Aoki, K., Kida, Y., Shimoyama, T., Sonoda, Y., Ueda, H.: SNFS248 (2004), http://www.rkmath.rikkyo.ac.jp/~kida/snfs248e.htm

Batch Pairing Delegation*

Patrick P. Tsang[1], Sherman S.M. Chow[2], and Sean W. Smith[1]

[1] Department of Computer Science
Dartmouth College
Hanover, NH 03755, USA
{patrick,sws}@cs.dartmouth.edu
[2] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu

Abstract. Pairing-based cryptography (PBC) has enabled the construction of many cryptographic protocols. However, there are scenarios when PBC is too heavyweight to use, such as when the computing devices are resource-constrained. Pairing delegation introduced in [19] provides a solution by offloading the computation to more powerful entities.

In this paper, we introduce the concept of, and construct several protocols for, *batch pairing delegation*, which offers significantly improved efficiency over multiple runs of state-of-the-art (non-batch) delegation protocols. We prove the security of our proposed protocols in the model we formalized for batch pairing delegation. Also, we have implemented our protocols in software for experimentation.

Moreover, we argue that the secure delegation of pairing computation, batched or not, requires different protocols depending on the semantic meaning of the pairings. We propose a taxonomy that classifies pairings into *seven* types to assist in choosing the right delegation protocol.

Finally, we propose a novel application of pairing delegation in trusted computing — we show how pairing delegation can be leveraged to build a secure coprocessor for pairing computation more cost-effectively.

1 Introduction

Pairing-based Cryptography. Since the first constructive uses of pairings over elliptic curves in cryptography such as tripartite key exchange [31] and identity-based encryption (IBE) [8,36], *Pairing-Based Cryptography (PBC)* has enabled for the first time secure and efficient construction of many novel cryptographic schemes such as attribute-based encryption [4,28], broadcast encryption [1,11], certificateless encryption [21,24], forward-secure encryption [14,40], searchable

* This research was sponsored in part by the National Cyber Security Division of the Department of Homeland Security, under Grant Award Number 2006-CS-001-000001, and by the NSF, under grant CNS-0524695. The views and conclusions do not necessarily reflect the views of the sponsors.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 74–90, 2007.
© Springer-Verlag Berlin Heidelberg 2007

encryption [12,38], aggregate signatures [9], short signatures [7], perfect non-interactive zero-knowledge (NIZK) [29] and multi-theorem NIZK [16].

Pairing Delegation. Pairing delegation, first introduced by Chevallier-Mames et al. [19], is a protocol during which an entity offloads the computation of pairings to another entity. In this paper, we refer to the entity who delegates the computation as the *Delegator*, or *Ron*, and the entity who actually does the work as the *Delegatee*, or *Ellen*.[1] The fact that computationally limited devices such as smartcards are slow and resource-constrained for computing pairings has been a major motivation for pairing delegation in [19,32]. By delegating the computation to more powerful machines such as PCs on which more efficient architecture and libraries for pairing computation exist, it becomes possible for those devices to execute various pairing-based cryptographic algorithms.

Security becomes a concern when the delegatee is not fully trusted by the delegator, and/or the two parties are communicating over an insecure channel. A pairing computation should be delegated as if the delegator computed the pairing himself. More precisely, neither the delegatee nor an eavesdropper should be able to learn anything about the pairing being computed, except perhaps the fact that the delegator is trying to compute a pairing. Furthermore, the delegator should be able to tell if he ends up with a correct answer to the computation of the pairing despite the adverse environment.

The only follow-up work (that the authors are aware of) gives several new protocol constructions with better efficiency and is due to Kang et al. [32].

Batch Processing. Under certain circumstances, processing tasks in batch rather than individually yields better efficiency. In batch signature verification [3], for instance, if verification returns valid, the verifier is assured that all signatures in the batch are valid with overwhelming probability. Otherwise, the verifier knows there is at least one bad signature. Further work on this topic studies efficient means to identify bad signatures [34,35]. Similarly, our idea of delegating pairing computation in batch tries to improve overall efficiency by delegating in batch rather than one by one independently.

Our Contributions. We make the following contributions in this paper:

– We introduce the concept of *batch pairing delegation* and provide *four* protocol constructions, which offer better efficiency than independently invoking state-of-the-art non-batch delegation protocols. We also obtain new non-batch delegation protocols. We formalize a new security model for batch pairing delegation and prove the security of our protocols in the model.
– We observe that the security requirements for pairing delegation protocols, batch or not, depend on the types of pairings being delegated. We thus propose a taxonomy of pairings and survey the literature extensively.
– We propose a novel use of pairing delegation in the arena of trusted computing and discuss some practical issues of pairing delegation.

[1] A mnemonic guide: Ron is the delegatoR and Ellen is the delegateE.

Paper Organization. In Section 2, we classify pairings into types and illustrate each type with extensive examples. We propose a security model for batch pairing delegation protocols in Section 3. Section 4 presents our batch pairing delegation protocol constructions and evaluates their efficiency. We show how pairing delegation can be used in trusted computing, and discuss some less apparent costs of pairing delegation in Section 5. Section 6 concludes the paper.

2 A Taxonomy of Pairing Types

Definition. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two additive cyclic groups and $\mathbb{G}_T$ be a multiplicative cyclic group, all of prime order p.[2] Suppose P and Q are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. A function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a *pairing* if it satisfies the following properties:

- *(Bilinearity.)* $e(xA, yB) = e(A, B)^{xy}$ for all $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_p$.
- *(Non-degeneracy.)* $e(P, Q) \neq 1$, where 1 is the identity element in $\mathbb{G}_T$.
- *(Efficient Computability.)* $e(A, B)$ can be computed in polynomial time for all $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$.

Depending on the pairing, $\mathbb{G}_1$ may or may not be the same as $\mathbb{G}_2$, and homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ may or may not be efficiently computable. While these variations have implications on many cryptographic schemes [26], they are not our concern as our delegation protocols work securely irrespective to them.

Taxonomy. We can classify pairings $e(A, B)$ into 16 types according to whether each of the points A and B is public or secret, and a constant or a variable, which is governed by the semantic meaning of the points at the protocol level.[3] While the type of a pairing is relatively unimportant when the pairing is computed locally, it makes a huge difference in how the pairing should be delegated due to different security requirements and optimization possibilities.[4]

Consequently, one must devise, and prove the security of, a delegation protocol with respect to a specific pairing-type. Also, it is important to understand the type of a pairing in a cryptographic protocol in order to select a secure delegation protocol. We therefore propose a taxonomy of pairings into types. In addition, the taxonomy indicates that some pairing-types are more utilized than the others (in the literature today), which may be used as a heuristic to evaluate the impact of an efficient delegation protocol.

[2] Recently, *composite-order groups* with a bilinear map have been constructed [10] and used (e.g. [11,12,13,16,29]). In this paper, we assume groups are of prime order for simplicity, even though our results remain valid for composite-order groups.

[3] For example, the decryption algorithms of many IBE schemes [8,27,36] require the computation of $e(S, U)$, where S is the decryption key and U is a part of the ciphertext. Thus, S is a secret constant while U is a public variable.

[4] For instance, the delegator does not need to hide a point from the outside if the point is public. Also, it may be possible for the delegator to pre-compute some of the operations if one of the points is a constant.

Note that not all 16 types require a distinct delegation protocol. Half of them are duplicates of the rest due to the symmetry between the roles played by points A and B in the protocols.[5] Also, there is no need to delegate when both A and B are constant. As a result, we propose a taxonomy of pairings into 7 types.

Type-SV2. Pairings of this type are such that both points are a secret variable. We call this the "general" type because intuitively a secure delegation protocol for this type should also work securely if one or both of the points are instead public, and/or a constant. To the authors' best knowledge, no cryptographic construct in the literature requires the computation of Type-SV2 pairings. Nevertheless, existing works [19,32] use delegation protocol for this general type as a basis to construct protocols for delegating pairings of other types.

Type-SVSC. Type-SVSC pairings have a secret variable point and a secret constant point. Their common usage is to prevent the leakage of partial knowledge about some secrets, exemplified in the following. Decryption in the inversion IBE scheme in [6] requires the computation of $e(A+rB, K)$, where A and B are from the ciphertext and (r, K) is the private key. When delegating the computation, one should treat $A + rB$ as a secret variable, or otherwise partial information about r would be leaked.[6] Another example is the ID-based key agreement protocol due to [18], wherein one has to compute the pairing of a secret key and a point derived from an ephemeral Diffie-Hellman secret exponent. Public knowledge of the latter point means partial information of such an exponent is leaked, which is not covered by the security guarantee in many key agreement protocols [22].

Few cryptographic schemes use Type-SVSC pairings. However, our delegation protocol for pairings of this type forms the basis for our other proposed protocols.

Type-SVPV. Pairings of this type are such that one point is a secret variable and the other is a public variable. We give several examples of their use here. In searchable encryption [12,38], the search gateway is delegated with different trapdoor for different keywords, so a pairing of secret variables (the trapdoors) and public variables (ciphertext) will be used. In trace-and-revoke broadcast system [11], the secret variable is a temporary key derived from the private key in a way depending on the set of legitimate decryptors, while the public variable comes from the ciphertext as usual. Pairings of this type also appear in the private ciphertext validity checking in [27], wherein the secret variable is determined by a function of the private key and the ciphertext while the public variable comes from the ciphertext. A less obvious example is an ID-based key agreement protocol due to Chow and Choo [22]. One might think that the point associated with the secret key is essentially an ID-based signature due to [15] and can be made public. However, the other point is also a public variable so it is the only secret knowledge for ensuring the confidentiality of the session key.

[5] This is true even for asymmetric pairings.

[6] It is not stated explicitly in [6] whether r in private key is just an auxiliary data for decryption that is safe to be publicly known.

Finally, we note that the some computations appeared as Type-SVPV can actually be replaced by Type-PVPC (the secret variable is replaced by a public constant) if the secret variable A is chosen by the one who computes the pairing. Instead of choosing A directly, one can choose $a \in_R \mathbb{Z}_p$ and set $A = aP$ instead. The required value for the pairing can be obtained from exponentiating a Type-PVPC pairing with the secret exponent a. Examples include the signcryption step in Chow et al.'s scheme [23], and a zero-knowledge proof for the correctness of the partial decryption in Baek and Zheng's ID-based threshold decryption [2].

Type-SVPC. Pairings of this type have a secret variable point and a public constant point, and are rarely used in the cryptographic constructs nowadays. Type-SVPC pairings appear when one wants to verify the private key obtained from the key generation center. An example is the ciphertext-policy attribute-based encryption in [4], where the secret variables come from the different components of the private key. Again, similar to Type-SVPV pairings, some pairings that appear to be Type-SVPC can be treated as pairings of two constants instead, e.g. in the proving step of Kurosawa and Heng's ID-based identification protocol [33] and in the signing step of Hess's ID-based signature [30].

Type-PVSC. Type-PVSC pairings have a public variable point and a secret constant point, and commonly appear in the decryption algorithm, in which the decryption key is the secret constant and the ciphertext contributes to the public variable. Examples include IBE [8,27,36], certificateless encryption [21,24], and other related encryption schemes such as attributed-based encryption [4,28], ID-based broadcast encryption [1] and forward-secure encryption [14,40]. Type-PVSC pairings also exist in some key agreement protocols [17,18].

Type-PV2. Pairings of this type pair up two public variables. They are present in many cryptographic constructs. In particular, they are commonly found in verification of signature schemes and ciphertext validity checks in encryption schemes. Examples include Boldyreva's multisignature and blind signature [5], Boneh et al.'s aggregate signature and ring signature [9], Boneh and Boyen's short signature [7], Chow's verifiable pairing [20], Dodis and Yampolskiy's verifiable random function (VRF) [25], the public ciphertext validity checking in [2,21], and Groth et al.'s witness-indistinguishable homomorphic proof commitments [29]. The multiplication of ciphertext in the doubly homomorphic encryption in [10] also uses this type of pairings. Type-PV2 pairings also appear in many cryptographic schemes involving an implicit tripartite key exchange [31]. The public variables come from the public keys and the proof/signature/ciphertext.

We note that one may treat the varying public key as a constant when batch-verifying signatures from the same signer, and hence classify the pairings as the Type-PVPC (see below) for better efficiency. However, if a point is a combination of a public key and a signature, this trick is not possible (unless a single pairing computation is split into two). An example is Cha and Cheon's scheme [15].

Type-PVPC. Pairings of this type are such that one of points is a public variable and the other is a public constant. They commonly appear in the encryption algorithm of IBE schemes based on full-domain hash [8,40], and the verification of pairing-based schemes such as Boldyreva's multisignature and blind signature [5], Boneh et al.'s aggregate signature, ring signature and verifiably encrypted signature [9], Boyen and Water's group signature [13], Chow's verifiable pairing [20], the VRF of Chase and Lysyanskaya [16] and Dodis and Yampolskiy [25], and Hess's ID-based signature [30]. In these examples, the public constant is either the master public key or a group generator, and the public variable comes from the identity of the decryptor or the signature respectively.

Summary. We end this section by noting that pairings of type PVSC, PV2 and PVPC are widely used by cryptographers today. Delegation protocols for these types can thus be considered very applicable. As we shall see, our proposed protocols cover two of these three types. Type-SV2 pairings, on the contrary, is not being used today. A delegation protocol for pairings of this type could be considered practically useless.

3 Security Model

We formalize the security requirements of batch delegation protocols for pairings in which one of the points is a constant, which are sufficient for our proposed protocols. The requirements can however be generalized for other pairing-types.

Let $A \in \mathbb{G}_1$ be the constant point, and $B_1, \ldots, B_n \in \mathbb{G}_2$ be the variable points to be paired up with A. The protocol's goal is to compute $\{e(A, B_i) : i \in \{1, \ldots, n\}\}$. Let $\mathcal{R}$ and $\mathcal{E}$ be two probabilistic polynomial time (PPT) algorithms modeling the Delegator and the Delegatee respectively. Let $out \overset{R}{\leftarrow} \mathcal{R}(in_1)_{\mathcal{E}(in_2)}$ denotes out is the output of $\mathcal{R}(in_1)$ in the interaction with $\mathcal{E}(in_2)$, where the inputs in_1 and in_2 may or may not be equal. Finally, $\mathcal{BDH}(k)$ denotes a instance generator for groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with orders 2^k and a pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Desired properties of a batch delegation protocols for pairings include:

Completeness. The Delegator $\mathcal{R}$ obtains $\{e(A, B_i) : i \in \{1, \ldots, n\}\}$ after interacting with an honest Delegatee $\mathcal{E}$.

Secrecy. Even a cheating Delegatee $\mathcal{E}$ cannot learn any information about A and $\{B_i : i \in \{1, \ldots, n\}\}$. The simulator is given A (respectively $B_1, \ldots, B_n$) if and only if A (respectively $B_1, \ldots, B_n$) is public.

We start by giving definition for the case in which A is public. The definition for any subset of $\{B_1, \ldots, B_n\}$ being public can be defined similarly. We need to define two more notations about simulatability:

- $Trans_B(param, A, B_1, \ldots, B_n)$ denotes a PPT algorithm outputting the transcript of communication between the interaction of the Delegator $\mathcal{R}(param, A, B_1, \ldots, B_n)$ and the Delegatee $\mathcal{E}(param, A)$.

- $Sim_B(param, A)$ denotes a PPT algorithm outputting a transcript simulating the one between the interaction of the Delegator $\mathcal{R}(param,\ A,\ B_1,\ \ldots,\ B_n)$ and the Delegatee $\mathcal{E}(param, A)$.

The protocol has secrecy if for all PPT adversaries $\mathcal{A}$, there exists a PPT simulator Sim such that $|\Pr[\mathbf{Game}_{\mathcal{A}}^{\mathrm{secr}_B}(k) = 1] - \frac{1}{2}|$ is negligible in k, over the random coins of all algorithms.

```
$\mathbf{Game}_{\mathcal{A}}^{\mathrm{secr}_B}(k)$
$param \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \overset{R}{\leftarrow} \mathcal{BDH}(k)$
$b \overset{R}{\leftarrow} \{0, 1\}$;
if $(b = 0)$
    then $\Gamma \overset{R}{\leftarrow} Trans(param,\ A,\ B_1,\ \ldots,\ B_n)$;
    else $\Gamma \overset{R}{\leftarrow} Sim(param, A)$;
$\hat{b} \overset{R}{\leftarrow} \mathcal{A}(param, A, \Gamma)$;
if $(b = \hat{b})$ then return 1 else return 0;
```

Correctness. The Delegator $\mathcal{R}$ can detect (with non-negligible probability) when the Delegatee $\mathcal{E}$ is cheating (i.e. the final result leads to a wrong value).

```
$\mathbf{Game}_{\mathcal{E}}^{\mathrm{corr}}(k)$
$param \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \overset{R}{\leftarrow} \mathcal{BDH}(k)$
$(t_1, t_2, \ldots, t_n) \overset{R}{\leftarrow} \mathcal{R}(param,\ A,\ B_1,\ \ldots,\ B_n)_{\mathcal{E}(param, A, B_1, \ldots, B_n)}$;
if $(\exists i \in \{1,\ \ldots,\ n\}, (t_i \neq e(A, B_i)))$ return 0; else return 1;
```

The protocol is correct if the probability that $\mathbf{Game}_{\mathcal{E}}^{\mathrm{corr}}(k) = 1$, for all $param$ generated by $\mathcal{BDH}$, for all $(A,\ B_1,\ \ldots,\ B_n) \in \mathbb{G}_1 \times {\mathbb{G}_2}^n$, and for all PPT adversary $\mathcal{E}$ (that may deviate from the protocol in arbitrary way), over the random coins of all algorithms, is non-negligible in k.

This game is basically saying that, for any cheating Delegatee $\mathcal{E}$ and for any $(A, \{B_i : i \in \{1,\ \ldots,\ n\}\})$, the Delegator $\mathcal{R}$ outputs either $\{e(A, B_i) : i \in \{1,\ \ldots,\ n\}\}$ or $\perp$, denoting invalid, except with negligible probability, even $(A, \{B_i : i \in \{1,\ \ldots,\ n\}\})$ is given to the cheating Delegatee $\mathcal{E}$.

Note that the definition of correctness in [19,32] does not specify whether A and/or B are known to the adversary (or a malicious delegatee). Such an ambiguity could lead to insecure delegation protocols. For example, we were able to come up with a delegation protocol by simplifying one of the protocols in [32] that is secure if and only if A is private. Our adversarial model *explicitly* equips the adversary with the knowledge of all points, which is stronger than what is expected since $\mathcal{E}$ is not given some of the points in some cases.

We also remark that even both points are public, a secure pairing delegation protocol is still necessary due to the correctness requirement.

4 Our Protocols

We present our construction of four protocols for delegating the computation of pairings in batch, each for a different type of pairings. In all four protocols, the interaction is between delegator Ron and delegatee Ellen during which

Ron, given a list of n pairs of points $\langle (A, B_1), \ldots, (A, B_n) \rangle$, tries to compute $\langle e(A, B_1), \ldots, e(A, B_n) \rangle$ with the help of Ellen. The common input and private input to each of the two parties vary from one protocol to another, but the distinction should be clear from the context. At the end of the protocol runs, Ron outputs either the correct answers, or $\perp$, which indicates a failure.

In our protocols, Ron is equipped with a secret constant $e(A, Q)$ for a random and private Q. The value of such a constant may be obtained through a one-time pre-computation either by Ron himself, or by a trusted third party.

All four proposed protocols are secure under the security model defined in Section 3. The security proofs can be found in Appendix A.

4.1 Our Four Constructions

Protocol-SVSC. This protocol delegates in batch the computation of n Type-SVSC pairings as follows.

1. *(Precompute.)* Ron picks $r_A, r_Q \in_R \mathbb{Z}_p$ and computes $\tilde{A} = r_A A$, $\tilde{Q} = r_Q Q$ and $e(\tilde{A}, \tilde{Q}) = e(A, Q)^{r_A r_Q}$. This step may be precomputed before knowing the values of B_i's.
2. *(Request.)* Ron sends to Ellen $\langle \tilde{A}, \tilde{B}_0, \tilde{B}_1, \ldots, \tilde{B}_n \rangle$, where $r_i, b_i, \in_R \mathbb{Z}_p$ and $\tilde{B}_i = r_i B_i$ for $i = 1$ to n, and $\tilde{B}_0 = \tilde{Q} + \sum_{i=1}^{n} b_i \tilde{B}_i$.
3. *(Respond.)* Ellen sends to Ron $\langle \alpha_0, \alpha_1, \ldots, \alpha_n \rangle$, where $\alpha_i = e(\tilde{A}, \tilde{B}_i)$ for all $i = 0$ to n.
4. *(Verify.)* Ron verifies if $\alpha_i \in \mathbb{G}_T$ for $i = 0$ to n and $e(\tilde{A}, \tilde{Q}) \prod_{i=1}^{n} \alpha_i^{b_i} = \alpha_0$.
5. *(Output.)* Ron returns $\perp$ if the above verification failed. Otherwise he returns $\langle \alpha_1^{1/r_A r_1}, \ldots, \alpha_n^{1/r_A r_n} \rangle$.

Protocol-SVPC. To batch delegate Type-SVPC pairings, follow the steps in Protocol-SVSC, except that r_A is set to 1 instead of a random element so that $\tilde{A}$ is equivalent to A. As a result, the computation and the transmission of $\tilde{A} = r_A A$ in Step 1 and 2 are eliminated.

Protocol-PVSC. To batch delegate Type-PVSC pairings, follow the steps in Protocol-SVSC, except that all r_i's are set to 1 instead of random elements so that $\tilde{B}_i$ is equivalent to B_i for all $i = 1$ to n. The steps related to of all $\tilde{B}_i$'s of the protocol are also eliminated.

Protocol-PVPC. To batch delegate Type-PVPC pairings, follow the steps in Protocol-SVSC, except that r_A and all r_i's are set to 1 so that $\tilde{A}$ is equivalent to A and $\tilde{B}_i$ is equivalent to B_i for $i = 1$ to n. As a result, the computation of $\tilde{A}$ in Step 1, the computation of all $\tilde{B}_i$'s in Step 2, the transmission of $\tilde{A}$ in Step 2, and the exponentiation of all α_i's in Step 5 of the protocol are all eliminated.

Remarks. In the last two protocols, all the points are public and we do not care secrecy. Nevertheless, a secure pairing delegation protocol is still necessary for the correctness requirement.

Each of our four batch protocols degenerates to a non-batch protocol if we set n to 1. We therefore obtain as a side-product four protocols for conventional pairing delegation. As will become clear in the next section, the degenerated version of our proposed protocols either performs better or is as good as the existing state-of-the-art protocols.

4.2 Performance Analysis

We now analyze the performance of the four protocols presented above in terms of both time and space complexities. Specifically, we contrast each of our four protocol constructions for delegating n pairing computations in batch to n parallel and independent invocations of traditional (non-batch) protocol for delegating pairings of the same type. We look at several aspects in our comparison, namely the size of communication, the online computational cost and total computational cost on the delegator and the total computational cost on the delegatee.

All four protocols selected for comparison are state-of-the-art in the literature in terms of efficiency. Two of them are due to Chevallier-Mames et al. [19] and the other two are due to Kang et al. [32] (see Table 1). It has been assumed in all their four protocols that the delegator already knows $e(X, Y)$ for some random X and constant Y before the protocol execution. Ron must therefore either compute $e(X, Y)$ himself or obtain the value from a trusted third party. While it is not explicitly suggested in [19,32] which should the case, we believe that the latter is much less favorable in practice.[7] Consequently, we assume, for the sake of a fair comparison, that it is Ron himself who computes $e(X, Y)$ in these protocols, by picking a $x \in_R \mathbb{Z}_p$ and calculating $X = xP$ and $e(X, Y) = e(P, Y)^x$, where $e(P, Y)$ is a constant already known to Ron. This adds 1 $\mathbb{G}_1/\mathbb{G}_2$ Scalar Multiplication (SM) and 1 $\mathbb{G}_T$ Exponentiation (EXP) to the offline computational costs of Ron in their protocols.

The comparison is detailed in Table 1. We make a few observation here. First, the size of communication is at least halved in all four cases. Hence, our protocols give a speedup in communication of a minimum of 2. Second, Ellen sees a speed up of 2 in all four cases because she has to compute only half the number of pairings using our protocols. Also, online computational costs on Ron are roughly the same except that our Protocol-SVPC is slightly worse than that in [32, §4.3]. Finally, the total computational costs on Ron are always smaller in all our protocols. Specifically, if we ignore the computation of $\mathbb{G}_1/\mathbb{G}_2$ point additions and $\mathbb{G}_T$ multiplications (as their costs are dominated by the costs of the computation of $\mathbb{G}_1/\mathbb{G}_2$ scalar multiplication and $\mathbb{G}_T$ exponentiation respectively), the speed up ranges from 1.3 to 4, depending on the relative speed of computing SMs and EXPs.

[7] The following must hold for the protocols to remain secure: the value of $e(X, Y)$ is correct, $e(X, Y)$ is known only to Ron, and Y is known only to Ron in cases where Y represents some secret of Ron's. These imply a trusted third party with high trustworthiness is required, the maintenance of which is costly.

Table 1. Performance comparison between our proposed protocols and multiple runs of state-of-the-art non-batch protocols when the batch size is n

Type	Protocol	Communication Complexities		Computation at Ron (#SMs,#PAs,#EXPs,#MULs)*		Computation at Ellen
		#rounds	(#$\mathbb{G}_1/\mathbb{G}_2$, #$\mathbb{G}_T$)	Online cost	Total cost	(#Pairings)
SVSC	[32, §4.2]	2	(4n,2n)	(n,n,4n,n)	(5n,n,5n,n)	2n
	Our Protocol-SVSC	**2**	**(n+2,n+1)**	**(2n,n,3n,n)**	**(2n+2,n,3n+1,n)**	**n+1**
SVPC	[32, §4.3]	2	(2n,2n)	(n,n,3n,n)	(3n,n,4n,n)	2n
	Our Protocol-SVPC	**2**	**(n+1,n+1)**	**(2n,n,3n,n)**	**(2n+1,n,3n+1,n)**	**n+1**
PVSC	[19, §6.2]	2	(4n,2n)	(n,n,3n,n)	(4n,n,4n,n)	2n
	Our Protocol-PVSC	**2**	**(n+2,n+1)**	**(n,n,3n,n)**	**(n+2,n,3n+1,n)**	**n+1**
PVPC	[19, §6.1]	2	(2n,2n)	(n,n,2n,n)	(2n,n,3n,n)	2n
	Our Protocol-PVPC	**2**	**(n+1,n+1)**	**(n,n,2n,n)**	**(n+1,n,2n+1,n)**	**n+1**

* PA and MUL denote $\mathbb{G}_1/\mathbb{G}_2$ point addition and $\mathbb{G}_T$ multiplication respectively.

4.3 Implementation and Experimentation

The fact that computationally limited devices such as smartcards are slow and resource-constrained for computing pairings has been a major motivation for delegation in [19,32]. By delegating the computation to more powerful entities such as PCs on which efficient libraries for pairing computation exist, it becomes practical for those devices to execute pairing-based cryptographic algorithms.

In this regard, pairing delegation makes sense only if it is faster for an entity to delegate the computation than doing the work itself. For instance, Scott et al. suggested recently in [37] that delegation might not offer any benefit because it does not offer significant gain in speed. In particular, the time for computing a pairing is comparable to the time for doing other underlying group operations.[8]

Nevertheless, one should not project from such a result, that was obtained from one experimental testbed with a particular set of parameter choices, and conclude that pairing delegation does not make sense in all scenarios. In fact, the relative speed of computing pairings and various underlying group operations can vary greatly depending on numerous factors ranging from the choices of curves and fields and their representation, to instruction sets, compilers and libraries in case of general-purpose processors, or architecture and level of parallelism in case of dedicated hardware coprocessor implementation. Pairing delegation speeds up pairing computation at the protocol level independent of the choice of any parameters mentioned above. We believe that the pairing delegation offers significant benefits in many real-world settings.

To gain some empirical data, we implemented our four proposed protocols in `C` using the PBC library[9] (version 0.4.7) for its elliptic-curve and pairing operations. The machine we used for the experiments was a Lenovo T60 laptop PC with an Intel dual-core 2GHz CPU and 1.5GB of Ram, running Ubuntu 6.10. Timing figures collected using such a platform serve as a heuristic of the lower bound on the efficiency gain using our batch delegation protocols because pairing computation has already been made very efficient on such a platform.

[8] This is indeed an exciting result because it demonstrates researchers' success in speeding up pairing computation and hence making PBC ever-increasingly practical.

[9] `http://crypto.stanford.edu/pbc/`

We did experiments to compare the latency of computing pairings locally (without any delegation) to that of using our four protocols, over various types of underlying elliptic curves.[10] In particular, we measured the sum of the time our protocols spent on performing the *Request* and the *Verify* steps. This measurement reflects the latency experienced by the delegator if we ignore the communication overhead and assume that the delegatee can compute pairings in relatively no noticeable time. Experiments were repeated 10 times using random input points over which timings were averaged. We used a batch size of 100.[11]

We calculated the speedups gained from using our protocols over computing pairings locally. For example, using a Type-F curve, it took 7.87s to compute the 100 pairings locally while a latency of 3.34s was measured when our Protocol-SVSC was used. The speedup was thus 7.87s/3.34s=2.34. Empirical results for other experimental parameters are summarized in Table 2. As shown by the existence of speedups that are less than 1, delegation did not always outperform local computation in terms of speed. This happens, for example, for curves over which computing a pairing and computing an SM and/or an EXP take comparable time. However, delegation did shorten the latency of pairing computation in the majority of cases and some of the speedups were significant.

Table 2. The speedup gained by using our proposed protocols over computing the 100 pairings locally using different types of curves as defined in the PBC Library

Curve	**Protocol-SVSC**	**Protocol-SVPC**	**Protocol-PVSC**	**Protocol-PVPC**
Type-A	0.75	**1.75**	0.73	**2.20**
Type-A1	**1.13**	**2.59**	**1.13**	**3.23**
Type-D225	0.49	**1.01**	0.49	**1.48**
Type-E	**1.17**	**3.22**	**1.20**	**3.60**
Type-F	**2.34**	**2.67**	**2.36**	**6.73**
Type-G	0.69	**1.29**	0.67	**2.04**

5 An Application Scenario

In this section, we propose a new application of pairing delegation and discuss some less apparent costs of delegation.

A Novel Application in Trusted Computing. An application for pairing delegation that has not been considered so far falls into the arena of trusted computing. Research in trusted computing strives to raise the trustworthiness of computing devices such as PCs by guaranteeing that they operate correctly and securely even under certain software and/or hardware attacks. To achieve this goal, trusted computing almost always uses some kind of tamper-resistant, tamper-evident, and/or tamper-responsive hardware. For example, the IBM 4758/4764 secure coprocessor [39] is a general-purpose PC built into a PCI card

[10] Please refer to the PBC Library for the characteristics of these curves.

[11] As shown earlier, the batch size has little effect on the timing of the delegator.

that is wrapped around in an armor made of layers of material capable of detecting tampering attempts. *Trusted Computing Group (TCG)*'s *Trusted Platform Module (TPM)*[12] takes a radically different approach—TPMs provide minimal tamper-resistant storage barely enough for storing a handful of cryptographic key material, leaving most of the operation outside the physical protection boundary. Being cheap enough to be deployed in all PCs was one of the design goals.

The Challenge. In general, we are faced with the following dilemma when building any trustworthy device, be it a general-purpose machine that runs arbitrary software, or a dedicated chip devised for a specific application—On one hand, we would like to put as much circuitry as possible into the *Trusted Computing Base (TCB)* to make the device more powerful. But on the other, it gets very pricey as the protection boundary grows due to increased complexity in configuration, maintenance and upgrade. Heat dissipation also places a practical limit on how powerful the hardware inside the TCB could possibly get.

A specific instance of the above challenge we would like to solve in this paper is *how to build a cost-effective pairing coprocessor with high assurance in physical security.* We would like to compute pairings correctly and securely despite the potential presence of various software and/or hardware attacks, because insecure pairing computation implies whatever cryptographic protocols for which these pairings are computed become insecure.

Our Solution. Pairing delegation provides a solution to the problem. One could build a secure coprocessor for pairing computation using two co-operating modules that are unbalanced in their size, power and trustworthiness. Specifically, one of the modules is lightweight but physically secured, which plays the role of the pairing delegator. The other is powerful but physically unprotected, and acts as the pairing delegatee, as shown in Figure 1. The security of pairing delegation guarantees that a pairing coprocessor built in this way operates correctly and securely even if only the small delegator module is physically armored. Consequently, such a coprocessor is lower in cost thanks to a smaller TCB, but still achieves good performance because the actual heavyweight pairing computation is now performed by the more powerful delegatee module.

Building a pairing coprocessor using the delegation approach as described above also allows more fine-grained engineering decisions for higher flexibility and better optimization. It is possible, for instance, that the delegator uses a multiplier module of smaller digital size for the underlying field arithmetics owing to limitation on space and/or cost while the delegatee does field multiplication in software on a multi-core PC for lower non-recurring engineering costs. This approach also permits upgrade, maintenance and reconfiguration of the delegatee accelerators without having to re-engineer the TCB.

Some Less Apparent Costs. Pairing delegation incurs additional costs that must be taken into account so as to correctly decide whether it is more

[12] https://www.trustedcomputinggroup.org/home

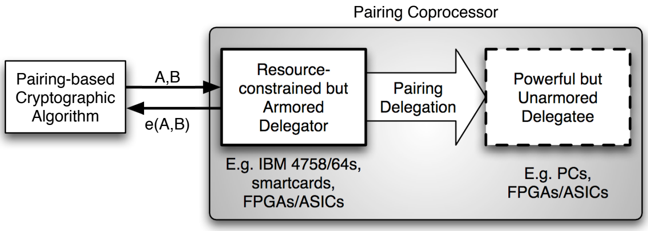

Pairing Coprocessor
Pairing-based Cryptographic Algorithm
A,B
e(A,B)
Resource-constrained but Armored Delegator
Pairing Delegation
Powerful but Unarmored Delegatee
E.g. IBM 4758/64s, smartcards, FPGAs/ASICs
E.g. PCs, FPGAs/ASICs

Fig. 1. An architecture for a cost-effective and trustworthy secure coprocessor for pairing computation that leverages pairing delegation

cost-effective to delegate the computation of pairings. Below we mention some of the less apparent costs that might get ignored.

Denial-of-Service (DoS) Attacks. Delegation protocols are inherently vulnerable to DoS attacks as the delegator needs cooperative help from the delegatee. An adversary can launch DoS attacks by corrupting the delegatee, or simply jamming the channel. Batch delegation protocols are even more vulnerable as corrupting the value of one pairing in the batch renders all the pairings useless.[13]

Timing-analysis Attacks. Cryptographic protocols are in general more susceptible to timing attacks than cryptographic algorithms due to the existence of communication events, the timing of which may be correlated to some internal secret values. Additional measures must be taken to compensate this extra attack surface opened up due to the delegation protocols' necessity to communicate.

Source of Randomness. Pairing delegation requires a cryptographically secure *Random Number Generators (RNG)*, while computing pairings locally does not. A secure RNG relies on special hardware to collect random physical events and incurs costs that vary depending on platform architectures.

Communication Costs. Communication could dominate the whole protocol execution time in the case when the communication channel has a low bandwidth, and/or a high latency. Delegation would thus be undesirable. Also, energy-constrained devices such as sensor nodes may prefer not to delegate because communication is very costly in terms of power consumption.

Latencies. Gains in efficiency by delegating pairings in batch grow with the size of the batch. However, computation latencies also grow with batch sizes, as the answer to the computation of any pairing in the batch won't be available until the delegatee has completed computing the whole batch. There is thus a tradeoff between efficiency and latencies. For instance, while batch protocols are useful in case of verifying aggregate signatures, they might be unsuitable if pairing computation occurs rarely, or answers to the computation are needed promptly.

[13] A potential solution would be to employ some error-detection techniques to identify the "bad" pairings in a batch.

6 Conclusions

In this paper, we have introduced the concept of batch pairing delegation and provided four protocol constructions, each for a different pairing-type. The security of our protocols has been proven under a new security model we formalized. We have implemented our protocols in software for experimentation, and shown that our batch delegation protocols are more efficient than both individually delegating pairing computation using the state-of-the-art non-batch protocols and computing the pairings locally without any delegation. Moreover, we have proposed how to build a secure coprocessor for pairing computation cost-effectively using delegation, and discussed some of the less apparent costs of delegation.

References

1. Abdalla, M., Kiltz, E., Neven, G.: Generalized key delegation for hierarchical identity-based encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, Springer, Heidelberg (2007)
2. Baek, J., Zheng, Y.: Identity-based threshold decryption. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 262–276. Springer, Heidelberg (2004)
3. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
5. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-Group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
6. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology (to Appear, 2008)
8. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
10. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
11. Boneh, D., Waters, B.: A collusion resistant broadcast, trace and revoke system. In: ACM Conference on Computer and Communications Security (CCS) (2006)

12. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
13. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
14. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. Journal of Cryptology (to Appear, 2007)
15. Cha, J.C., Cheon, J.H.: An identity-based signature from Gap Diffie-Hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2002)
16. Chase, M., Lysyanskaya, A.: Simulatable VRFs with applications to multi-theorem NIZK. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, Springer, Heidelberg (2007)
17. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. International Journal of Information Security 6(4), 213–241 (2007)
18. Chen, L., Kudla, C.: Identity based authenticated key agreement protocols from pairings. In: IEEE Computer Security Foundations Workshop, pp. 219–233 (2003)
19. Chevallier-Mames, B., Coron, J.-S., McCullagh, N., Naccache, D., Scott, M.: Secure delegation of elliptic-curve pairing. Cryptology ePrint Archive, 2005/150
20. Chow, S.S.M.: Verifiable pairing and its applications. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 170–187. Springer, Heidelberg (2005)
21. Chow, S.S.M., Boyd, C., Nieto, J.M.G.: Security-mediated certificateless cryptography. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 508–524. Springer, Heidelberg (2006)
22. Chow, S.S.M., Choo, K.-K.R.: Strongly-secure identity-based key agreement and anonymous extension. In: TSDM 2000. LNCS (to appear, 2007)
23. Chow, S.S.M., Yiu, S.-M., Hui, L.C.K., Chow, K.P.: Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 352–369. Springer, Heidelberg (2004)
24. Dent, A.W.: A survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211 (2006)
25. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
26. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006) http://eprint.iacr.org/
27. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
28. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. ACM CCS, pp.89–98 (2006)
29. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)

30. Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
31. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
32. Kang, B.G., Lee, M.S., Park, J.H.: Efficient delegation of pairing computation. Cryptology ePrint Archive, Report 2005 /259, (2005), http://eprint.iacr.org/
33. Kurosawa, K., Heng, S.-H.: The power of identification schemes. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 364–377. Springer, Heidelberg (2006)
34. Pastuszak, J., Michatek, D., Pieprzyk, J., Seberry, J.: Identification of bad signatures in batches. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 28–45. Springer, Heidelberg (2000)
35. Pastuszak, J., Pieprzyk, J., Seberry, J.: Codes identifying bad signature in batches. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 143–154. Springer, Heidelberg (2000)
36. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: Symposium on Cryptography and Information Security (SCIS) (2000)
37. Scott, M., Costigan, N., Abdulwahab, W.: Implementing cryptographic pairings on smartcards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)
38. Shi, E., Bethencourt, J., Chan, H.T.-H., Song, D.X., Perrig, A.: Multi-dimensional range query over encrypted data. In: IEEE Symposium on Security and Privacy, pp. 350–364 (2007)
39. Smith, S.W., Weingart, S.: Building a high-performance, programmable secure coprocessor. Computer Networks 31(8), 831–860 (1999)
40. Yao, D., Fazio, N., Dodis, Y., Lysyanskaya, A.: ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. ACM CCS, pp.354–363 (2004)

A Security Proofs

Completeness is trivial. Secrecy is also easy to see. The random choice of r_A and/or the random choices of $r_1, \cdots, r_n$ (depends on which of A and B's are public) each time makes the view of the delegatee follows the same distribution.

The following proves that all of our four proposed delegation protocols satisfy the requirement of correctness. *A single proof* is sufficient since its crux depends on the randomness of r_Q that is present in all our proposed protocols.

Note that we give an *information theoretic* argument here. Moreover, all our protocols are single-round, so they are secure against "concurrent attacker".

Since $e(P, Q)$ is a generator of $\mathbb{G}_T$, we assume without loss of generality that the adversary returns the values of $\{\alpha'_i : i \in \{0, 1, \cdots, n\}\}$ in the following forms:

$$\alpha'_i = e(\tilde{A}, \tilde{B}_i)e(P, Q)^{\beta_i}, \forall i \in \{0, 1, \cdots, n\}, \text{ where } \beta_i \in \mathbb{Z}_p, \forall i \in \{0, 1, \cdots, n\}$$

To break the correctness, $\exists j \in \{1, \cdots, n\}$ such that $\beta_j \neq 0$. Note that any other β_is, in particular β_0, can take any value arbitrarily.

Suppose $\alpha_i = e(\tilde{A}, \tilde{B}_i), \forall i \in \{0, 1, \cdots, n\}$, which means $\alpha_0 = e(\tilde{A}, \tilde{Q}) \prod_{i=1}^{n} \alpha_i^{b_i}$. For verification not to return $\perp$, we have the following equation holds:

$$\begin{aligned}
\alpha'_0 &= e(\tilde{A}, \tilde{Q}) \prod_{i=1}^{n} {\alpha'}_i^{b_i} \\
\alpha_0 \cdot e(P, Q)^{\beta_0} &= e(\tilde{A}, \tilde{Q}) \prod_{i=1}^{n} (\alpha_i \cdot e(P, Q)^{\beta_i})^{b_i} \\
e(P, Q)^{\beta_0} &= \prod_{i=1}^{n} e(P, Q)^{\beta_i \cdot b_i} \\
\beta_0 &= \sum_{b=1}^{n} b_i \beta_i \\
b_j \beta_j &= \beta_0 - \sum_{i \in \{1, \cdots, n\} \setminus \{j\}} b_i \beta_i
\end{aligned}$$

From $\tilde{B}_0 = r_Q Q + \sum_{i=1}^{n} b_i \tilde{B}_i$, as long as r_Q is randomly chosen each time, there are always p possibilities for b_j. Without loss of generality, we assume $\alpha'_j \neq e(A, B_j)^{1/r_A r_j}$, i.e. $\beta_j \neq 0$. With b_j being an unknown, the probability that the above equality holds is at most $1/p$. Note that we made no assumption on other α'_is for $i \neq j$ are correct or $(A, \{B_i : i \in \{1, \cdots, n\}\})$ being unknown to the cheating delegatee. However, we do assume Q is kept secret from the delegatee, otherwise partial information about r_Q will be leaked when the delegatee knows A and/or $\{B_i : i \in \{1, \cdots, n\}\}$.

Botnet Traffic Detection Techniques by C&C Session Classification Using SVM

Satoshi Kondo[1] and Naoshi Sato[2]

[1] Trend Micro Incorporated, Shinjuku MAYNDS TOWER 27F 2-1-1 Yoyogi,
Shibuya-ku, Tokyo, Japan
satoshi_kondo@trendmicro.co.jp

[2] Institute of Information Security, 2-14-1 Tsuruya-chou, Kanagawa-ku,
Yokohama, Japan
sato@iisec.ac.jp

Abstract. Bots, which are new malignant programs are hard to detect by signature based pattern matching techniques.

In this research, we focused on a unique function of the bots the remote control channel (C&C session). We clarified that the C&C session has unique characteristics that come from the behavior of bot programs. Accordingly, we propose an alternative technique to identify computers compromised by the bot program for the classification of the C&C session from the traffic data using a machine learning algorithm support vector machine (SVM). Our evaluation resulted in 95% accuracy in the identification of the C&C session by using SVM. We evaluated that the packet histogram vector of the session is better than the other vector definitions for the classification of the bot C&C session.

1 Introduction

We are facing a new type of malware called bot[1] whose attacks are increasing. The bots spread widely and infect many computers through the Internet. They are designed for distributed attacks such as DDoS and port scanning, for reconnaissance of networks and computers, and as a SPAM mail dispatcher. These bot activities are different from existing malware such as viruses and worms. Their main activities do not focus on attacking the infecting host; instead, the hosts are compromised to use them in a distributed attack platform.

The bot program creates a communication and control channel to the attacker and it is used to conduct over then thousands of the compromised computers by them. This communication and control channel is called the command and control (C&C) channel. The attacker controls bots using this channel to execute distributed attacks and other activities. That is called botnet, and it is causing serious security problems. Until the bot receives a command from the attacker, it stays dormant in the compromised host. This makes bots harder to detect than other viruses and worms.

The other peculiar feature of bots is that they have generated many variants in a short time, like an explosion. Attacker communities share the source code

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 91–104, 2007.
© Springer-Verlag Berlin Heidelberg 2007

of bots and breed them to be more powerful. It is easy to generate more bots by sharing the source codes. However, the attackers do not have many programming skills, so they find and use the bot program development kit to make their own bots.

Furthermore, the bot program uses the packer techniques to encrypt and archived itself with a self-extraction and decryption function to generate unique binary files from a single source. This combination generates many kinds of unique binary files of the bot program, which are spread throuhgh the Internet.

1.1 Related Works

There are several approaches to detect bots and computers compromised by them. Here, we summarize them and address the related issues.

Signature- and Rule-based Detection. For bot detection, the most popular method is signature-based pattern matching. This approach has been used typically for malware detection. In this approach, it is necessary to prepare signature data for each malware if its binary file is different. Therefore, signature-based pattern matching cannot find the malware program if the signature is not available. As described above, bots are generated with many short-term valiants and each variant is not widely propagated to the Internet. Therefore, it is hard for a security vendor to capture entire bot programs. As a result of not capturing entire bot programs, signature-based pattern matching technique has a low bot detection rate[2].

On the other hand, a different approach based on IDS/IPS such as snort[3] is used to detect the bots. This alternative method uses packet-matching rules to detect those security attacks using the well-known security vulnerabilities. This approach can detect any variant of the bot when it uses the same security vulnerability for infection. This approach depends on the knowledge of the security vulnerability and exploits; therefore, it cannot detect any new security attack, called a zero day attack. We cannot prepare for the zero day attacks until we have analyzed and identified conceptual or real attacks.

Behavior and Activity Detection. James et al.[4] proposed an algorithm to detect bot communication channels on an IRC[5] server using the work weight of the channel members. They defined the work weight as the ratio of the count of TCP control packets to all TCP packets. The TCP control packet is enabled by some TCP flag bit such as RST. If the compromised computer performs a port scan or exhibits some network attacking behavior, the work weight increases. Therefore, the scanning activities by malwares such as bots are identified. Based on this work weight, James et al. monitored an IRC server channel and detected which IRC channel is used by bots to communicate. However, the compromised computer cannot be detected until the bot is activated by the attacker to execute network attacking activities. The infection activities of bots uses not only security vulnerabilities but other methods as well, such as email, P2P file sharing, uploading to web sites and instant messenger services.

Anirudh et al.[6] used the activities to lookup a DNS-based black list (DNSBL) by the botnet owner. One of the purposes of the bots is to dispatch SPAM mail. The DNSBL is commonly used to block the server that relays SPAM. The mail server refers the list to identify the SPAM relay server and rejects mail transfer from it. Once it is listed in the DNSBL, the mail server does not transfer the mail to the other mail relay server and hence blocks incoming SPAM mail from the mail relay. The study found that the bot owners periodically look up the DNSBL to check if their own compromised computer is listed. The botnet owners can confirm, whether the compromised computer still has the capability of sending SPAM mail by checking the DNSBL. This unique activity of checking the DNSBL is useful in identifing the compromised computer which is used for dispatching SPAM mail. However, it could not be applied to detect the compromised computer infected by any other kind of bot.

C&C Session Detection. Christopher[7] report the bots C&C session detection using snort. This approach analysis IRC packet payload and define detection rule for specific bot command strings. It can detect well known bots, but cannot detect any variant of bots that uses different command strings. The attacker easy to decieve these detection rules to modify their own bot programs.

Carl et al.[8] attempted two stage identification approach to detect the bots C&C session using machine learning algorythms . The first stage, they split the IRC session from whole traffic data. The second stage, identify the bots C&C session from classified IRC session data. They achieved the low false-positive rate (7.89%) for the bots IRC session detection using Naïve Bayes. However, they prepared single kind of bot program to collect the bots C&C sessions for training data.

1.2 Motivation

We addressed the issue of bot detection in the previous section. The motivation for this research is to find a generic approach to detect the computer compromised by any kind of bot; one that does not depend on any single instance of the bot program and specific activities. Therefore, we focused on the communication channel used by the attacker to control compromised computers remotely. This is a generic activity of bots. We can identify the computer compromised by a bot from the source IP address of the C&C sessions, if we can identify the C&C sessions from the entier network traffic.

2 Analysis of Bot C&C Session

First, we explain the collection of the malware dataset which we used to learn the characteristics of the bot C&C session traffic. Then, we discuss the characteristics of the bot C&C session learned from our analysis.

2.1 Dataset

We used the collection of malware data set which is captured by our honeypot system. The honeypot system uses nepenthes[9] collect bots from the Internet. It is operated by the bot analysis team of the Institute of Information Security. In this research, we used 2161 unique binary files that were captured by this system. The malware collection was scanned by ClamAV[10] antivirus tool (version 0.88.2 and signature file number 2416) and the result are shown in Table 1.

The antivirus program identified 1473 files as bots and 366 unknown files. (”Unknown” means ClamAV does not have a signature to identify the bot.) Exploit.DCOM.Gen is detected as generic DCOM exploit code in the file. It cannot identify any existing malware, but most of them are a viruses, worms, or new bot programs that are not identified exactly by the signature.

We captured all IP packets during execution of each malware on the sandbox environment for traffic analysis. We executed a malware under Windows XP (with no service pack applied) in VMware[11] for 3 minutes. All packets from/to this sandbox environment were captured and stored as files in tcpdump format.

We found that some of the malwares were corrupted. Some of them tried to connect to an IRC server as the C&C server; they gave the error that the server was shoutdown or not reachable. In other cases, the domain name of the server was deleted from the DNS entry.

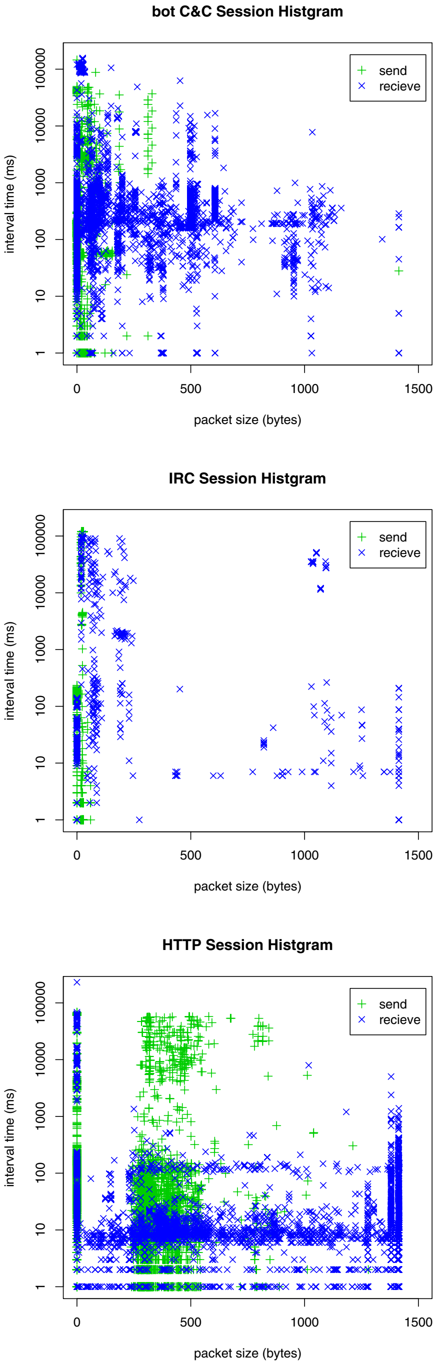
Table 2 shows the result of malware execution. By hand-analysis, we identified 957 active bots and 1229 C&C server sessions. Those bots accessed to 97 unique servers. Some of them were running on the same machine. That machine was compromised by attackers and multiple C&C servers were installed and configured. We used these 1229 sessions as bot C&C sessions for analysis of characteristics and used for classification. More details of their analysis and examination of the classification are given in the following section.

Table 1. Contents of Malware Specimen Collection

malware type	sub type	num.	percentage
SDBot	123	598	27.67%
MyBot	197	539	24.94%
PoeBot	19	243	11.24%
IRCBot	18	93	4.30%
Others	84	205	9.49%
Exploit.DCOM.Gen		117	5.41%
Unknown		366	16.94%
total		2161	100.00%

2.2 Analysis of Bot C&C Session Characteristics

We analyzed the captured packet data to identify specific characteristics of C&C sessions. In this analysis, the payload of the packets and protocol header information such as protocol types, source and destination IP addresses, and port



bot C&C Session Histgram

IRC Session Histgram

HTTP Session Histgram

Fig. 1. Packet Histogram of the Session

Table 2. Detection of Active Bots and C&C Servers

Active bot programs (C&C session detected)	957
Unknown (ClamAV not detected as a malware)	67 (7%)
Total C&C Sessions	1229
Unique Servers (pair of IP addresses & port number)	97
Unique IP addresses	71

number was not inspected. However, the protocol identification that checks some unique protocol signature such as command strings is easy to invalidate by encryption or modification of the protocol. In addition, there are privacy issues if we apply this technique to an ISP or some backbone network to check the packet payload data. Therefore, we focused on minimum information such as the packet size and packet interval time.

The traffic management research field contains many reports on the classification of sessions and traffic. Andrew et al.[12] used Bayesian analysis to classify Internet traffic. They categorized applications in several groups by their traffic characteristics. They achieved 95% accuracy to classify the traffic to categorized groups by 248 discriminators per flow data. Laurent et al.[13] proposed a technique that relies on the observation of the first five packets of a TCP connection to identify the application. It has a better detection rate even when using minimal discriminators such as first five packets.

Based on these reports, we used background information such as packet payload size and packet interval time. These discriminators should be as minimal as possible and easy to collect on the network. Our challenge was to find any identical characteristics between the C&C session and the IRC chat session. Both sessions are based on same protocol with minimum differences.

Fig.1 shows the packet histogram plot with respect to the packet payload data size and the interval time from the previous packet.

We can see unique characteristics of the C&C sessions that are compared with HTTP sessions and IRC chat sessions; however, these are derived from simple information. For example, in the Web session plot, the send packet size was plotted around 200 through 500 bytes. It is represented that request command of HTTP protocol from the client. The receive packet size was plotted and had maximum packet payload size near 1500 bytes. Most packet interval times where short.

Even though both sessions were using same protocol, the IRC protocol, we can see some differences between them. This means some information and characteristics do not depend on the protocol but on usage and behavior.

The application behavior and functionality appear in that information.

In contrast, the bot program uses the IRC servers installed into the compromised computers as the C&C servers. Therefore the compromised computer has lower bandwidth and processing performance than typical open relay IRC servers. This also effect the interval time of the packet sequence. The attackers maintain their own IRC servers for bot control to prevent the operator from shuting down the communication channels to block the bot activities. Attackers need to use some customized version of the IRC server that is optimized for handling to over 10,000 bots as an IRC client. There are many functional trade-offs and requirements to support some specific features of the bot activities. Those characteristics are difficult to modify and mimic for other applications. Some functionality and architecture of the specific activities of the bots appear similar. It is the unique characteristics of the session for bot programs that have similar functionality.

Therefore, we can use this information for the classification and identification of the bot C&C session.

3 Experiment of C&C Session Classification

3.1 Definition of the Feature Vector

To evaluate the effectiveness and accuracy of the session classification using the vector definition as discriminators, we defined three kinds of vectors for session classification (Fig. 2). These are session information vector, packet sequence vector, and packet histogram vector.

Session Information Vector. The vector is defined as total receive packet numbers, total receive packet data size, total send packet numbers, total send packet data size and session time. This session information vector is generated using these five values for each session from the packet capture data file. It is the most simple vector definition.

Packet Sequence Vector. The packet sequence vector consists of the packet size, and packet interval time of the first 16 packets from the session established. It does not include packet payload size of 0 bytes such as control packets like SYN packets.

The difference in protocol appears at the beginning after the session established. For most protocols, the first few packets represent the protocol negotiation stage, therefore there are unique characteristics shown by each protocols.

Packet Histogram Vector. The packet histogram vector is the histogram data by packet payload size and packet interval time in the session. We divided the packet payload size in 16 ranges of 100 bytes each, and the packet interval time in 10 ranges on a log scale. Send and receive packets are separated by count histogram data. The send packets were defined as packets outgoing from the sandbox environment and the receive packets are defined as incoming packets.

This vector has the 320 vector values; 160 vectors for the send packets histogram and the rest for the receive packets histogram.

3.2 Dataset for Experimentation

We prepared 4 datasets for the C&C session classification. We used them as a set to create three experimental vector data.

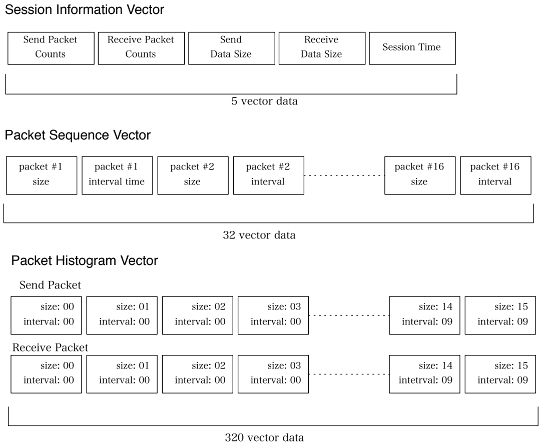


Fig. 2. Feature Vector Definitions

Training Dataset. We used 1029 sessions for the training dataset as the bot C&C sessions. Also, we used 6581 sessions as a background noise which were not identified as the C&C sessions during execution of malware on the sandbox environment. We labeled them as 'bot' and 'other' for training of the classifier.

Testing Dataset. This dataset includes 200 sessions of the C&C session and 800 sessions from background sessions.

IRC Chat Dataset. We prepared a typical IRC chat sessions dataset to verify the classification functionality between a C&C session and a normal IRC chat session.

HTTP Dataset. This dataset includes 132 HTTP sessions which are generated by normal web browsing operation by us within 1 hour. We used this dataset as typical session data of the network traffic.

3.3 Support Vector Machine

The support vector machine (SVM)[14] is known as a better machine learning algorithm for two class discrimination. It shows good accuracy in the classification of voice dictation, image recognition, and other pattern-matching applications. The SVM has better classification functionality and processes the high dimension of the vector data well for training and classification. Therefore, we used the SVM for the C&C session classification. In the experimentats, we evaluated the performance of the SVM and compared it with the other algorithms such as Naïve Bayes and k-Nearest Neighbor (k-NN)[15].

3.4 Experiment Environment

In this experimentats, we used R[16] to verify the classification functionality by SVM. The SVM module is provided as the e1071 package[17] for R; it is based on LIBSVM[18] implementation.

4 Results

In this section, we show the result of the C&C session classification by SVM using each vector definition.

4.1 Session Information Vector

Table 3 shows the result of C&C session classification by the session information vector. It correctry detects 863 sessions as bot C&C session from 1029 sessions in the training data set and detects 162 sessions as bot C&C session from 200 sessions in the testing data set. In other words, the detection rate for C&C sessions of the training dataset is 83.87% and 81.00% for the test dataset. That was a good classification for the bot C&C session using simple vector data to represent the session characteristics. However, the false-positive rate is higher (46.15%) for the IRC chat session. It misclassified the normal IRC chat session as the C&C session. The session information vector data does not reveal the differences between the C&C and normal IRC chat session, because they are using the same protocol.

4.2 Packet Sequence Vector

Table 4 shows the classification result using the packet sequence vector. In this result, all of the C&C sessions in the training dataset were correctly identified as such. However, there is an 82.00% false-negative results for classification of the C&C sessions in the testing dataset. That is, it does not have better classification for the non-training dataset.

4.3 Packet Histogram Vector

Table 5 shows the classification result using the packet histogram vector. The result was better than the other two vector definitions. It classified the C&C session in the training dataset and testing dataset well. The false-positive rate is 0.03% in the training dataset; the other data had no false-positive result. The false-negative rate is 2.62% in the training dataset and 5.00% in the testing dataset.

Comparing with the signature based detection, this has the better false-negative rate result (7% in the Table 2).

Table 3. M_I : The Detection Rate of the Session Data Vector

result	training set		testing set		IRC	HTTP
	bot	other	bot	other		
bot	863	4	162	2	12	1
other	166	6577	38	798	14	131
total sessions	1029	6581	200	800	26	132
detection rate	83.87%		81.00%			
false-positive		0.06%		0.25%	46.15%	0.76%
false-negative	16.13%		19.00%			

Table 4. M_S : The Detection Rate of the Packet Sequence Vector

result	training set		testing set		IRC	HTTP
	bot	other	bot	other		
bot	1029	0	36	0	0	0
other	0	6581	164	800	26	132
total sessions	1029	6581	200	800	26	132
detection rate	100.00%		18.00%			
false-positive		0.00%		0.00%	0.00%	0.00%
false-negative	0.00%		82.00%			

4.4 Comparison of SVM with Other Classification Algorithms

We compared the result of C&C session classification using SVM with other classification algorithms. We used the same dataset and a three feature vector definition data format for Naïve Bayes and k-Nearest Neighbor (k-NN).

Fig. 3 shows that we recognized that SVM is better than the other algorithms for classifying the ession information. SVM provides a better result than the other algorithms for the false-positive rate (Fig. 3). The Naïve Bayes misclassified all sessions as the C&C session in all feature vector data format. The result by k-NN is similar to SVM except for the false-positive rate.

As Table 6 shows, the processing speed of the session classification using SVM was faster than other algorithms. The testing machine was Linux kernel 2.6.15 running on Intel Pentium M 2GHz CPU and 1Gbyte of memory. The classification processing by SVM took less than 1 ms per session. On the other hand, Nav̈e Bayes and k-NN took more than ten times longer to classify a session with the packet histogram vector. In particular, the processing took a long time at the session sequence vector and the packet histogram vector. In contrast, the feature vector does not significantly affect the processing speed by SVM. The result shows better scalability for high dimension vector data such as the packet histogram vector.

The training processing cost was expensive in the SVM. However, SVM needs to be trained only at the beginning and it can be preprocessed. The total processing of the SVM was faster than other algorithms. k-NN does not need training for the classification; therefore, it uses all training data for the classification to

Table 5. M_H : The Detection Rate of the Packet Histogram Vector

result	training set		testing set		IRC	HTTP
	bot	other	bot	other		
bot	1002	2	190	0	0	0
other	27	6579	10	800	26	132
total sessions	1029	6581	200	800	26	132
detection rate	97.38%		95.00%			
false-positive		0.03%		0.00%	0.00%	0.00%
false-negative	2.62%		5.00%			

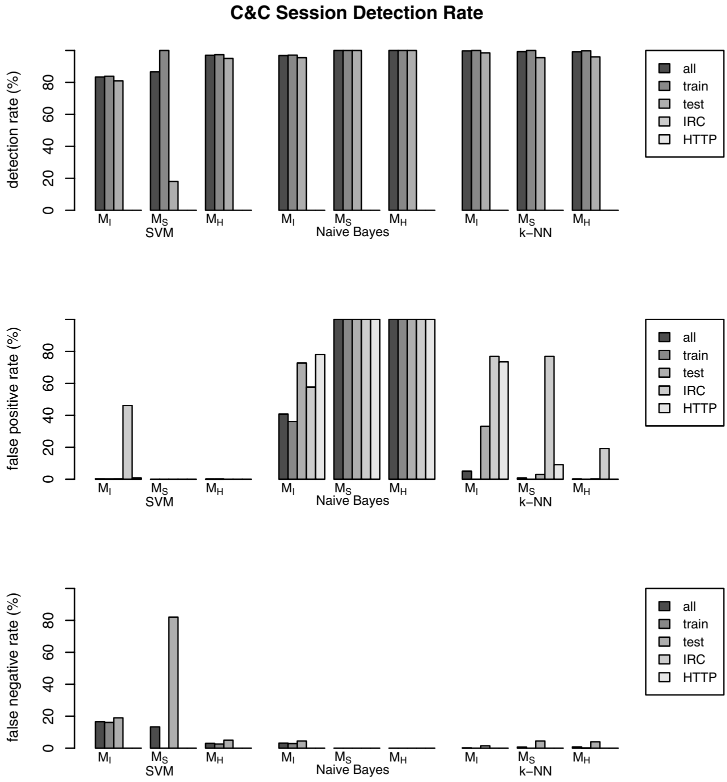


Fig. 3. Comparison of the C&C Session Detection Rate

calculate the distance between the target session vector data and training sessions vector datas for each session. The k-NN needs to optimize the training dataset to minimize the processing cost, but for unknown session data, it is difficult to simultaneously reduce the processing cost and maintain better accuracy of the classification.

Table 6. Processing Time for Training and Classification

Vector Definitions	SVM	Naïve Bayes	k-NN
Session Information Vector			
Total (s)	3.03	6.85	1.62
Training (s)	2.15	0.04	–
Classification (ms/session)	0.10	0.78	0.18
Session Sequence Vector			
Total (s)	47.46	50.66	88.36
Training (s)	38.60	0.72	–
Classification (ms/session)	1.01	5.70	10.08
Packet Histogram Vector			
Total (s)	6.73	159.42	393.91
Training (s)	5.41	1.99	–
Classification (ms/session)	0.15	17.96	44.93

5 Discussion

We discuss here the contribution of this research and the issue, to be verified in future work.

5.1 Training Dataset

We prepared the sample dataset generated from the sandbox environment. It does not include any other sessions belonging to user activities that appear in the real network traffic data. Those session data are effective for C&C session classification accuracy in providing a contrast between the characteristics of C&C sessions and other sessions. We predict that those session data provide better optimization of the support vector after training on SVM. However, it can have a negative effect on detection rate if those sessions have characteristics similar to the C&C session.

5.2 Feature Vector Definition

The detection result was strongly affected by feature vector definition from session information. In particular, it creates false-positive rates for a non-C&C session. This means we have to choose better definitions to represent characteristic differentiation between C&C sessions and other sessions.

In our research, we identified that the packet histogram vector provides a better result than the others. There may be some better vectors for the session classification.

5.3 Deceiving the Bots C&C Session Characteristics

The attacker can modify the C&C session characteristics to deceive our detection technique. For example, bots can emulate regular chat sessions and messages and mixed control command and messages into them. However, the attacker installs C&C server on the computer compromised by bots to construct the botnets to avoid to identified and shutdown. There are differences network behavior between open IRC server and these installed bots C&C servers for example band width and system performance. It will affect the packet histgram data and still we have a chance to detect them. However, we have to consider these counter approach to compromised our detection technique.

5.4 Application to the Various of C&C Session Protocols

In this research, we targeted the C&C sessions that are based on IRC protocols. In the facts, most of bots use IRC protocols, and some of them use modified protocol commands and encrypt the command and control messages.

Our approach in applicable to these bots, but some bots using other protocols for command and control exist.

These different protocol-based C&C sessions appear to have different traffic characteristics. However, They would have similar behavior that comes from the objectives and motivation of bots and functionality of botnet.

6 Conclusion

We conclude that our alternative approach is valid for detecting the computer compromised by bots. The difference in characteristics of the bot C&C sessions are useful for session classification, even though we used limited information such as the packet size and packet interval time. This difference was related to the functionality of the bot program; therefore it appeared for all bot programs.

We showed that the feature vector definition greatly influenced the identification accuracy, and using the packet histogram among the definitions of three feature vectors, we obtained a 95% detection rate for the non-training dataset. In particular, it had minimum false-positives and false-negatives in classifying C&C sessions and normal IRC chat sessions, which both use the same IRC protocol. Compared with other algorithms, SVM showed better accuracy in the classification of the C&C session and better scalable performance.

However, there are several issues need to be considered in the above. In particular, the counter approach of our proposed detection technique and targeting the non-IRC protocol based C&C sessions are required to concern in the next step from this research.

Acknowledgment

In this research, we used the collection of malware dataset by the bot research team in Tanaka Laboratory in the Institute of Information Security. We express our deep thanks to the team members.

References

1. Barford, P., Yagneswaran, V.: An Inside Look at Botnets. In: Special Workshop on Malware Detection, Advances in Information Security, Springer, Heidelberg (2006)
2. Nepenthes Development Team: http://nepenthes.mwcollect.org/stats:scannertest Available from http://nepenthes.mwcollect.org/stats:scannertest
3. M. Roesch: Snort: Lightweight intrusion detection for networks. In: 13th Systems Administration Conference (LISA'99), pp. 229–238. USENIX Associations (1999)
4. Binkley, J.R., Singh, S.: An Algorithm for Anomaly-based Botnet Detection. In: Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI), pp. 43–48 (July 2006)
5. Oikarinen, J., Reed, D.: Internet Relay Chat Protocol. RFC1459, Internet Engineering Task Force (1993)
6. Ramachandran, A., Feamster, N., Dagon, D.: Revealing Botnet Membership Using DNSBL Counter-Intelligence. In: Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI), pp. 49–54 (July 2006)
7. C.W. Hanna: Using Snort to Detect Rogue IRC Bot Programs. Technical report, (October 2004)
8. Livadas, C., Walsh, B., Lapsley, D., Strayer, T.: Using Machine Learning Techniques to identify botnet traffic. In: Proceedings of 2nd IEEE LCN Workshop on Network Security (November 2006)
9. Nepenthes Development Team: Nepenthes - Finest Collection. Available from http://nepenthes.mwcollect.org/
10. ClamAV project: ClamAV. Available from http://www.clamav.net/
11. VMware Inc.: VMware workstation. Software available at http://www.vmware.com/
12. Moore, A.W., Zuev, D.: Internet Traffic Classification using Bayesian Analysis Techniques. In: SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 50–60 (2005)
13. Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., Salamatian, K.: SIGCOMM Comput. Commun. Rev. Number 36(2), 23–26 (2006)
14. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
15. Fix, E., Hodges, J.: Discriminatory analysis: Nonparametric Discrimination: Consistency Properties. Technical report 21-49-004, USAF School of Aviation Medicine (1951)
16. R Development Core Team: R: A Language and Environment for Statistical Computing. (2005) http://www.R-project.org ISBN 3-900051-07-0
17. Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A.: A: The e1071 Package (2006), Available at http://cran.r-project.org/src/contrib/Descriptions/e1071.html
18. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. Software (2001), available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

A Global Authentication Scheme for Mobile Ad-Hoc Networks*

P. Caballero-Gil[1] and C. Caballero-Gil[2]

[1] Dept. Statistics, Operations Research and Computing. University of La Laguna.
38271 La Laguna. Tenerife. Spain
pcaballe@ull.es

[2] Dept. Informatics and Systems. University of Las Palmas de Gran Canaria. 35017
Las Palmas de Gran Canaria. Gran Canaria. Spain

Abstract. This work proposes a new global authentication system for Mobile Ad-hoc Networks. The component algorithms are designed in a self-organizing way so that most needs of this sort of networks are covered. In particular, characteristics such as adaptation to the varying topology of the network, open availability of broadcast transmissions, and strong access control have received special attention when defining the new scheme. The described protocol is based on the cryptographic paradigm of Zero-Knowledge Proofs. In this paper the design is thought for the Hamiltonian Cycle Problem, but it might be easily adapted to other NP-complete graph problems.

Keywords: Authentication, Access Control, MANETs.

1 Introduction

Confidentiality, integrity and authentication are the three main security aspects that have to be taken into account when designing a secure network. Among them, authentication, which guarantees the proper identities of nodes, is the most remarkable one because the other security characteristics depend totally on the right authentication of entities.

Authentication is usually based on weak schemes of maximum-disclosure proofs with secret time-invariant passwords [14]. Their major security concern is possible eavesdropping and subsequent replay of secret passwords. Two well-known solutions to this security problem exist. The simplest of both methods uses variable passwords, whereas the strongest schemes are minimum-disclosure proofs. The protocol here proposed combines both concepts in order to define an authentication scheme specifically thought for Mobile Ad-hoc Networks.

Mobile Ad-hoc NETworks (MANETs) are autonomous networks formed by mobile nodes that are free to move at will. The set of applications for MANETs is diverse, ranging from small, static networks that are constrained by power sources, to large-scale, mobile and highly dynamic networks. Conventional wired

* Research partially supported under Project SEG2004-04352-C04-03.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 105–120, 2007.
© Springer-Verlag Berlin Heidelberg 2007

networks mainly use a globally trusted Certificate Authority (CA) for solving the authentication problem. However, authentication in MANETs is in general much more difficult than that in wired networks due to several reasons such as limited physical protection of broadcast medium, frequent route changes caused by mobility, lack of a structured hierarchy, etc [12].

Many authentication protocols have been recently proposed for ad-hoc networks [1] [7] [11]. On the one hand, the paper [1] states the need for an authentication management architecture for ad-hoc networks. On the other hand, both papers [7] and [11] propose two different solutions. However, the first one is based on RSA signature, which conducts to the problem of public key certification, while the second solution works well just for short-lived MANETs.

In general, one of the most elementary approaches found in the bibliography uses a Trusted Third Party (TTP) to guarantee the validity of all nodes identities, so that every node who wants to join the network has to get a certificate from the TTP. A second identification paradigm that has been used in wireless ad-hoc networks is the notion of chain of trust [8]. A third typical solution is location-limited authentication, which is based on the fact that most ad-hoc networks exist in small areas and physical authentication may be carried out between nodes that are close to each other. The special nature of ad-hoc networks, where most applications are collaborative and group-based, suggests that such traditional approaches to node identification may not be always appropriate. Consequently, the design of a scheme that fulfils all the requirements for this type of networks continues being considered an open question.

This work proposes a different type of scheme based on the established cryptographic primitive of Zero-Knowledge Proofs (ZKPs), which provide an elegant solution to the problem of self-organized node authentication for MANETs. Until now very few publications have mentioned the proposal of authentication systems for ad-hoc networks using ZKPs [6] [2] [13], and none of them has dealt with the related problem of topology changes in the network. A recent ZKP-based hierarchical proposal for MANETs related with the one proposed here was described in [4], where two different security levels were defined through the use of a hard-on-average graph problem, and no topology changes were considered.

This work is organized as follows. The following section provides a complete description of the new proposal is given, including general aspects, notation and specific details about network initialization, node insertion, access control, proofs of life and node deletion.Assumptions on the scheme and security are commented in Section 3. A performance analysis is provided in Section 4. Finally, some conclusions and open questions complete the paper.

2 Proposal

The following sub-sections give, respectively, an overview of the proposal, a description of the used notation and specific details about network initialization, node insertion, access control, proofs of life and node deletion.

2.1 Overview

The proposal has been designed as an authentication scheme for group membership because when a node wants to be part of the network, it has to be previously authorized by a legitimate node. According to the authors of [10], in any group member authentication protocol it is necessary to provide robust methods to insert and to delete nodes, as well as to allow the access only for legitimate members of the group. For that reason, not only the ZKP used for access control is described later, but also the upgrade procedures associated to insertions and deletions are carefully defined. The procedure to delete nodes in this paper is based on the fact when a node is too long (according to a parameter previously agreed by the members of the network) disconnected of the network, a deletion of such a node is carried out.

The access control described below is based on the general scheme of Zero-Knowledge Proof introduced in [3], for the particular case of the Hamiltonian Cycle Problem (HCP). A hamiltonian cycle of a graph is a cycle that visits each vertex exactly once and returns to the starting vertex. Determining whether such cycles exist in a graph is the Hamiltonian Cycle Problem, which is NP-complete. Such a problem was chosen for our design mainly because the upgrade of a solution due to an insertion or a deletion of a vertex in the graph does not demand a great computational effort. Such operations will be frequent in our implementation due to the high dynamism of MANETs. Anyway, similar schemes might be described based on different NP-complete graph problems where the actualization of a solution after single changes on the graph is easy. Such is the case of Vertex Cover, Independent Set or Clique Problems, for instance.

One of the key points for the correct operation of the proposed scheme is the use of a chat application through broadcast that makes it possible for legitimate on-line nodes to send a message to all on-line users of the network. Such an application allows publishing all the information associated to the upgrade of the network. Although secrecy is not necessary for chat messages that are broadcast because they are useless for illegitimate nodes, since that information is necessary for updating authentication information, it is required that only on-line legitimate nodes of the network may execute the chat application.

The information received through the chat application during an interval of time must be stored by each on-line node in a FIFO queue. Such data stored by each on-line node allow the updating of the authentication information both for it and for all the off-line legitimate nodes whose access is authorized by that on-line node. The length of such a period is an essential parameter because it states both the maximum off-line time allowed for any legitimate node, and the frequency of broadcasts of proofs of life. Consequently, such a parameter should be previously agreed among all the legitimate nodes of the network.

A generic life-cycle of a MANET has three major phases as shown in Figure 1. Initialization is the first phase, where each initial member of the original network is securely provided, either off-line or on-line, with a secret piece of information.

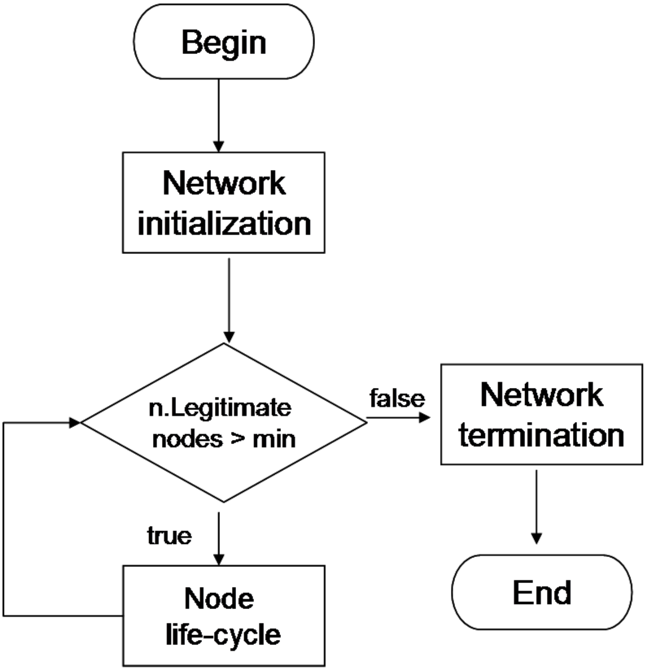


Fig. 1. Network Life-Cycle

The knowledge of the secret network key will be used during access control in order to prove the nodes eligibility to access protected resources or to offer service to the network.

When the initialization phase is completed, the initial legitimate nodes are ready to participate in the network, so node life-cycle starts (see Figure 2). The access control process is where a legitimate node proves its membership to an on-line node of the network. These legitimate nodes must demonstrate knowledge of the secret network key by using a challenge-response scheme.

Once the legitimate node access to the on-line state in the network is allowed, such a legitimate node gets full access to protected resources such as the chat application, and may offer services such as the insertion of new nodes. The secret network key is continuously being updated according to the development of the network, so the secret key of a legitimate node expires if this node is off-line too long. In such a case, the legitimate node would have to be re-inserted by an on-line legitimate node if it wants to enter the network again.

Since in our proposal the secrecy of the network key is provided by the difficulty of the HCP, and the number of legitimate nodes is an influential parameter in such a difficulty, as soon as the number of legitimate nodes is too small, the network termination is carried out and the life-cycle of the network ends.

A remarkable aspect of our proposal is that no possible adversary is able to succeed to steal any meaningful information even if it reads all the information

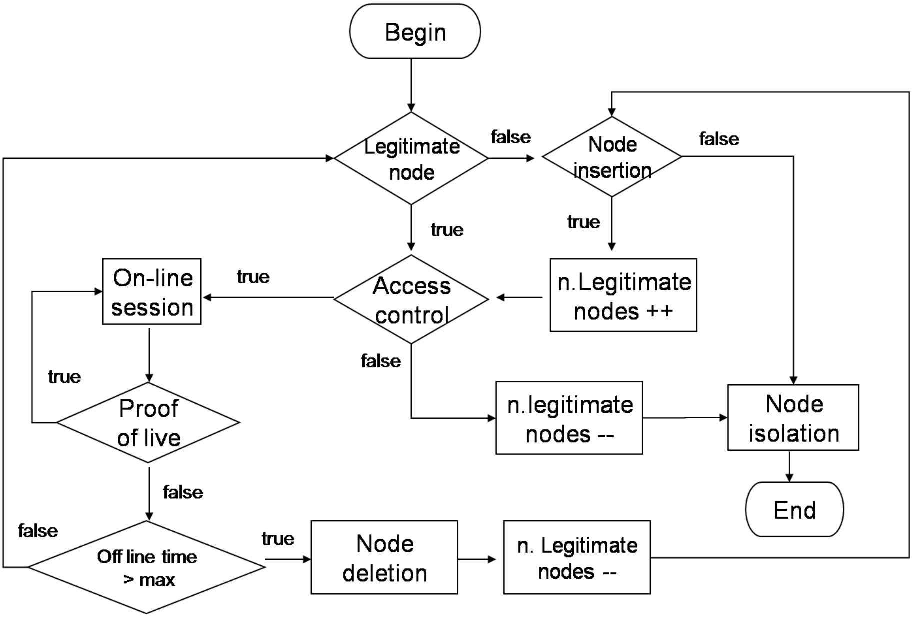


Fig. 2. Node Life-Cycle

published through the chat application, or if it eavesdrops all the information exchanged between a prover legitimate node and a verifier legitimate node during an access control.

2.2 Notation

In this section we give basic notations that are used throughout the proposal.

- $G_t = (V_t, E_t)$ denotes the undirected graph used at stage t of the network life-cycle.
- $v_i \in V_t$ represents both a vertex of the graph and a legitimate node of the network.
- $n = |V_t|$ is the order of G_t, which coincides with the number of legitimate nodes of the network.
- $N_{G_t}(v_i)$ denotes the neighbours of node v_i in the graph G_t.
- $\Pi(V_t)$ represents a random permutation over the vertex set V_t
- $\Pi(G_t)$ denotes the graph isomorphic to G_t corresponding to the permutation $\Pi(V_t)$.
- $c \in_r C$ indicates that an element c is chosen at random with uniform distribution from a set C.
- HC_t designates the hamiltonian cycle used at stage t.
- $\Pi(HC_t)$ represents the hamiltonian cycle HC_t in the graph $\Pi(G_t)$.

- $N_{HC_t}(v_i)$ denotes the neighbours of node v_i in the hamiltonian cycle HC_t.
- S and A stand for the supplicant and the authenticator, respectively, both during an insertion phase and during the execution of a ZKP-based access control.
- $S \rightleftharpoons A$ symbolizes when node S contacts A.
- $A \leftrightarrow S : information$ means that A and S agree on $information$
- $A \overset{s}{\rightarrow} S : information$ means that A sends $information$ to S through a secure channel.
- $A \overset{o}{\rightarrow} S : information$ means that A sends $information$ to S through an open channel.
- $A \overset{b}{\rightarrow} network : information$ represents when A broadcasts $information$ to all on-line legitimate nodes of the network.
- $A \overset{b}{\leftrightarrow} network : information$ represents a two-step procedure where A broadcasts $information$ to all on-line legitimate nodes of the network, and receives their answers.
- h stands for a public hash function.
- T denotes the threshold length of the off-line period for legitimate nodes.

2.3 Network Initialization

Such as it happens in most access control schemes for MANETs [5] [15], the proposed protocol requires the definition of an initialization phase where the secret information associated to the process of identification is generated and distributed within the initial network. This initialization phase consists in the definition of the graph used for the development of the protocol, jointly by all the original members of the network. Furthermore, the initialization phase implies that each legitimate member will know an initial and jointly generated solution of the HCP in such a graph.

In our proposal, as in trust graphs [9], the set of vertices of the graph corresponds exactly to the set of nodes of the real network during the whole life-cycle of the network. Consequently, the initialization process starts from a set V_0 of n vertexes corresponding to the nodes of the initial network. Furthermore, each vertex sub-index may be used as ID (IDentification) for the corresponding node. The first step of the initialization process consists of generating jointly and secretly a random permutation Π of such a set. Once this generation is completed, each legitimate node should know a hamiltonian cycle HC_0 corresponding exactly to such a permutation. Finally, the partial graph formed by the edges corresponding to such a hamiltonian cycle HC_0, is completed by adding n groups of $\frac{2m}{n}$ edges, producing the initial edge set E_0. Each one of these n groups of edges must have endvertex v_i, $i = 1, 2, ..., n$, and be randomly generated by the node v^i. Note that the cardinality $\frac{2m}{n}$ of those groups of edges must be large enough so that the cardinality of the resulting edge set $|E_0| = m$ guarantees the difficulty of the HCP in the graph G_0.

```
Initialization Algorithm
  Input: $V_0$, with $|V_0| = n$
    1. The $n$ nodes of the network generate jointly, secretly and randomly the
       cycle $HC_0 = \Pi\,(V_0)$.
    2. Each node $v_i \in V_0$ builds the set $N_{G_0}(i) = \{\{v_j \in_r V_0\} \cup N_{HC_0}(i)\}$ with
       $|N_{G_0}(i)| = \frac{2m}{n}$.
    3. Each node broadcasts $v_i \xrightarrow{b} \mathit{network} : N_{G_0}(i)$
    4. Each node merges $E_0 = \bigcup_{i=1,2,\ldots,n} \{(v_i, v_j) : v_j \in N_{G_0}(i)\}$
  Output: $G_0 = (V_0, E_0)$, with $|E_0| = m$
```

Once the construction of the initial instance of the problem has been carried out by means of the contribution of all the nodes that are part of the network, each node will know a hamiltonian cycle in the resulting $\frac{2m}{n}$-regular graph. From then on, each time a new user S wants to become a member of the network, it has to contact a legitimate member A in order to follow the insertion procedure explained in the following section.

2.4 Node Insertion

Let us suppose that we are at stage t of the network life-cycle when a user S contacts a legitimate member A of the network to become a member of the network. Once S has convinced A to accept its entrance to the network, the first step that A should do is to assign to S the lowest vertex number v_i not assigned to any node in the vertex set V_t. Afterwards, A should broadcast such an assignment to all on-line legitimate nodes of the network in order to prevent another simultaneous insertion with the same number, and receive their answer. If A receives less than $n/2$ answers, she stops the insertion procedure because the number of nodes that are aware of the insertion is not large enough. Otherwise, A chooses the corresponding upgrade of the secret hamiltonian cycle HC_t by selecting at random two neighbour vertexes v_j and v_k in order to insert the new node v_i between them, chooses at random a set of $\frac{2m}{n} - 2$ nodes in V_t such that none of them are neighbours in HC_t, and broadcasts the set of neighbours $N_{G_{t+1}}(v_i)$ of S in the new graph G_{t+1} to all on-line legitimate nodes of the network.

Each time a node receives an updating of the graph, it should secretly update the corresponding hamiltonian cycle by using the information provided in order to identify the unique position in the cycle where the new node can be inserted according o the new edge set E_{t+1}. In this way, it will be able to easily update the secret network key by simply inserting the vertex v_i between the vertexes v_j and v_k. At the same time the authenticator node A must send the supplicant node S both the graph G_{t+1} in an open way, and the hamiltonian cycle HC_{t+1} through a secure channel.

```
Insertion Algorithm
  Input: At stage $t$ a supplicant node $S$ wants to become a member of the
  network.
    1. $S \rightleftharpoons A$ and node $S$ convinces node $A$ to accept its entrance to the
       network.
    2. $A$ assigns to $S$ the vertex number $v_i$ such that $i = min\{l : v_l \notin V_t\}$
    3. $A$ broadcasts $A \overset{b}{\leftrightarrow} network : v_i$
    4. – If $A$ receives less than $n/2$ answers, she stops the insertion procedure.
       – Otherwise:
         (a) $A$ chooses at random $\{v_j \in_r V_t, v_k \in_r N_{CH_t}(v_j)\}$
         (b) $A$ chooses at random $N_{G_{t+1}}(v_i) = \{v_j, v_k\} \cup \{w_1, w_2, ..., w_{\frac{2m}{n}-2} \in_r$
             $V_t$ such that $\forall w_{l_1}, w_{l_2} : w_{l_1} \notin N_{CH_t}(w_{l_2})\}$
         (c) $A$ broadcasts $A \xrightarrow{b} network : N_{G_{t+1}}(v_i)$
         (d) Each on-line node computes $V_{t+1} = V_t \cup \{v_i\}$, $E_{t+1} = E_t \cup$
             $N_{G_{t+1}}(v_i)$ and $HC_{t+1} = \{HC_t \setminus (v_j, v_k)\} \cup \{(v_j, v_i) \cup (v_i, v_k)\}$
         (e) $A$ sends openly $A \xrightarrow{o} v_i : G_{t+1}$
         (f) $A$ sends securely $A \xrightarrow{s} v_i : HC_{t+1}$
  Output: The supplicant node $S$ is a legitimate member of the network.
```

2.5 Access Control

If a legitimate member of the network S that has been off-line or out-of-coverage from stage t wants to connect on-line to the network at stage r, its first step should be to contact a legitimate on-line member A. Afterwards, A should check whether the off-line period of S is not greater than T. In this case, S has to be authenticated by A through a ZKP of its knowledge of the secret solution HC_t on the graph G_t.

The aforementioned ZKP begins with the agreement between A and S on the number of iterations l of the ZKP. From there on, in each iteration, S will choose a random permutation $\Pi_j(V_t)$ on the vertex set that will be used to build a graph $\Pi(G_t)$ isomorphic to G_t. The hash value of that permutation $h(\Pi_j(V_t))$ and of the hamiltonian cycle in the graph $h(\Pi_j(HC_t))$ are then sent to A. When this information is received by A, it chooses a bit b_j at random ($b_j \in_r \{0,1\}$). Depending on the selected value, S will provide A with the image of the hamiltonian cycle through the isomorphism, or with the specific definition of the isomorphism. In the verification phase, A will check that the received information was correctly built.

Once the authentication of supplicant S has been successfully carried out, the authenticator A gives S the necessary information to have full access to protected resources such as the chat application.

```
Access Control Algorithm
  Input: At stage $r$ a supplicant node $S$ that has been off-line from stage $t$
  wants to connect on-line to the network.
    – $S \rightleftharpoons A$
    – $S$ sends openly $S \xrightarrow{o} A : G_t$
    – $A$ checks whether $t \leq r - T$
      • if $t \leq r - T$ then $S$ is not authenticated
      • otherwise:
          ∗ $A$ and $S$ agree $A \leftrightarrow S : l$
    * $\forall j \in \{1, 2, \ldots, l\}$
      1. $S$ chooses $\Pi_j(V_t)$ and builds $\Pi_j(G_t)$ and $\Pi_j(V_t)$, isomorphic
         graph to $G_t$ and correspondent hamiltonian cycle, respectively
      2. $S$ sends openly $S \xrightarrow{o} A : \{h(\Pi_j(V_t)), h(\Pi_j(HC_t))\}$
      3. $A$ chooses the challenge $b_j \in_r \{0, 1\}$
      4. $A$ sends openly the challenge $A \xrightarrow{o} S : b_j$
        (a) If $b_j = 0$ then $S$ sends openly $S \xrightarrow{o} A : \{\Pi_j(G_t), \Pi_j(HC_t)\}$
        (b) If $b_j = 1$ then $S$ sends openly $S \xrightarrow{o} A : \Pi_j$
      5. $A$ verifies
        (a) that $\Pi_j(HC_t))$ is a valid hamiltonian cycle in $\Pi_j(G_t)$, if
            $b_j = 0$
        (b) that the hash function $h$ on the result of $\Pi_j$ on $G_t$ produces
            $h(\Pi_j(G_t))$, if $b_j = 1$
    * if $\exists j \in \{1, 2, \ldots, l\}$ such that the verification is negative, then $S$
      is isolated.
    * otherwise $A$ sends securely $A \xrightarrow{s} S$ : the necessary information
      to have full access to protected resources of the network.
  Output: Node $S$ is connected on-line to the network.
```

2.6 Proofs of Life

All on-line legitimate nodes have to confirm their presence in an active way. Such a confirmation is carried out every certain interval of time of length T so that each on-line node must broadcast a proof of life to all on-line legitimate nodes of the network.

If some insertion happens during such a period, a proof of life of every on-line legitimate node will be distributed together with the broadcast necessary for the insertion procedure. If no insertion happens during the period, the first node that has to prove its life starts a proof-of-life broadcast. During such a broadcast every node adds its own proof of life to the broadcast so that when the broadcast reaches the last node, a broadcast back starts so that when the starting node receives the proofs of life of all on-line legitimate nodes, it rebroadcasts them.

```
Proof-of-Life Algorithm
  Input: At stage $t$ node $A$ is an on-line legitimate node of the network of the
  network.
    – $A$ initializes its $clock = 0$ just after its last proof of life
    – if $clock > T$ then
        1. $A$ broadcasts $A \overset{b}{\leftrightarrow} network : A's\ proof\ of\ life$
        2. • If $A$ receives less than $n/2$ proofs of life as answers to her broad-
             cast, she stops her proof of life and puts back her clock.
           • Otherwise: $A$ broadcasts $A \xrightarrow{b} network : Received\ proofs\ of$
             $life$
  Output: At stage $t + 1$ node $A$ continues being an on-line legitimate node of
  the network of the network.
```

2.7 Node Deletion

The deletion procedure is mainly based on the confirmation of the active presence of on-line legitimate nodes through their proofs of life. Each node should update its stored graph by deleting all those nodes that have not sent any proof of life after a period T. This fact implies that each node that has not proven its life will be deleted from the network, and the corresponding vertex will be deleted from the graph and from the hamiltonian cycle.

Node deletions are explicitly communicated to all on-line legitimate nodes in the second step of broadcasts of proofs of life. In this way, any node that is off-line in that moment will be able to update its stored graph as soon as it gets access to the network.

```
Deletion Algorithm
  Input: At stage $t$ a node $v_i$ is an off-line legitimate node of the network of the network.
    – $A$ initializes her $clock = 0$
    – if $clock > T$ then
        1. $\forall v_i \in V_t$: $A$ checks $v_i$'s proof of life in $A$'s FIFO queue
        2. $A$ updates $V_{t+1} = V_t \setminus \{v_i \in V_t$ with no proof $\}$
        3. $A$ updates $E_{t+1} = E_t \setminus \{(v_i, v_j) : v_i \in V_t$ with no proof, $v_j \in N_{G_t(v_i)}\} \cup \{(v_j, v_k) : v_j, v_k \in N_{HC_t(v_i)}\}$
        4. $A$ updates $HC_{t+1} = HC_t \setminus \{(v_j, v_i), (v_i, v_k)\} \cup (v_j, v_k) : v_i \in V_t$ with no proof, $v_j, v_k \in N_{HC_t(v_i)}$
    – If $A$ was the starter of the broadcast used for the $v_i$'s deletion, $A$ adds this information to the second step of the proof-of-life broadcast: $A \xrightarrow{b} network : v_i$ is deleted.
  Output: At stage $t + 1$ the node $v_i$ has been deleted both from the network and from the graph.
```

This way to proceed guarantees a limited growth of the graph that is used in authentication, and at the same time, allows that always legitimate nodes of the network correspond exactly to vertexes in that graph. Apart from this, it is remarkable the fact that thanks to this procedure the recovery of legitimate members of the network that have been disconnected momentarily due to a shortcut of the network is possible, if such a shortcut does not last too much (i.e. if it is lesser than T).

3 Assumptions and Security Analysis

This proposal assumes initially the ideal environment where all legitimate nodes are honest and where no adversary may compromise a legitimate node of the network in order to read its secret stored information. Such assumptions are well suited as a basic model in order to decide under which circumstances the designed authentication scheme is applicable to MANETs. For instance, a possible adaptation of the proposal in order to avoid those hypothesis could be the consideration of a threshold scheme for every step of the scheme, so that every proof of life, insertion, access control or deletion should be done by a group of on-line nodes each time. In this way, a dishonest node would not affect the correct operation of the network.

It is also clear that the proposal inherits inherent problems of the distributed trust model such as the important necessity that legitimate nodes cooperate. Consequently, it is advisable that some scheme to stimulate node cooperation is used in conjunction with the proposal.

Finally, another requirement of the scheme is the necessary establishment of a secure channel for the insertion procedure. However, that aspect may be easily fulfilled thanks to the fact that most wireless devices communicate with each other via Bluetooth wireless technology.

With respect to possible attacks, due to the lack of a centralized structure, it is natural that possible DOS (Denial Of Service) attacks have as their main objective the chat application. In order to protect the scheme against this threat it must be assured that chat messages, although are publicly readable, may be only sent by legitimate on-line members of the network. Another important aspect related to the use of the chat application is the necessary synchronization of the on-line nodes, so a common network clock is necessary. this requirement has been implemented during simulations through the chat application.

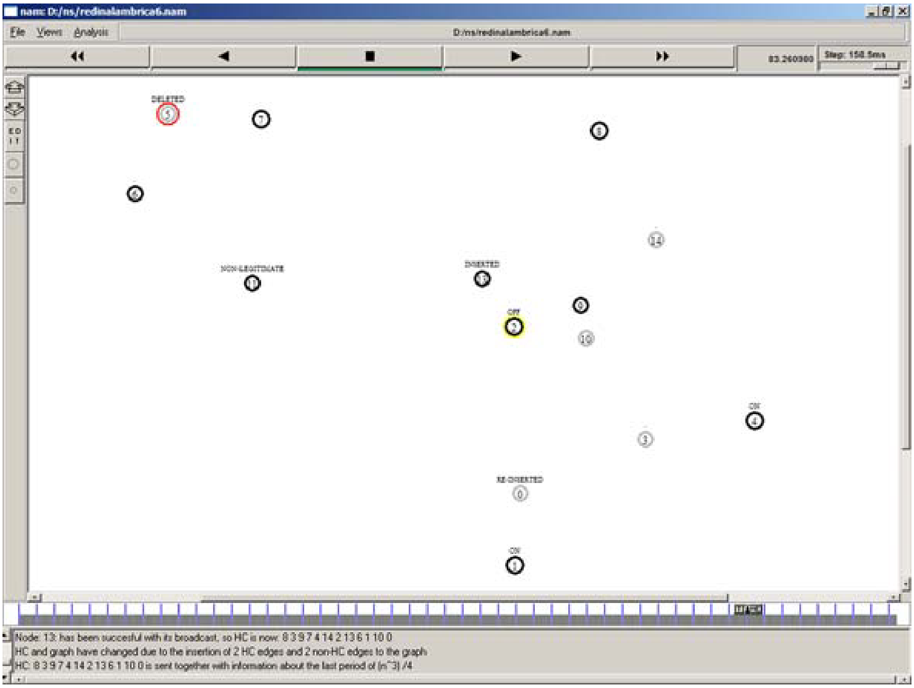
MANETs are in general vulnerable to different threats such as identity theft (spoofing) and the man-in-the-middle attack. Such attacks are difficult to prevent in environments where membership and network structure are dynamic and the presence of central directories cannot be assumed. However, our proposal is resistant to spoofing attacks because access control is proved through a ZKP that makes useless the reading of any information published through the chat application or sent openly during an access control. On the other hand, the goal of the man-in-the-middle attack is either to change a sent message or to gain some useful information by one of the intermediate nodes. Again the use of ZKPs in our protocol implies that reading any transferred information does not reveal any useful information about the secret, so changing the message is not possible since only legitimate nodes whose access has been allowed can use the chat application.

Another active attack that might be especially dangerous in MANETs is the so-called Sybil attack. It happens when a node tries to get and use multiple identities. The most extreme case of this type of attacks is the establishment of a false centralized authority who states the identities of legitimate members. However, this specific attack is not possible against our scheme due to its distributed nature. In our scheme, the responsibility of controlling general Sybil attacks will be shared among all the on-line nodes. If an authenticator node detects that a supplicant node is trying to get access to the network by using an ID that is yet being used on-line, such access control must be denied and the corresponding node must be isolated. The same happens when any on-line node detects that an authenticator node is trying to insert a new node to the network with a new ID, and such a node has yet assigned a vertex ID. Again, such insertion must be denied and the corresponding supplicant node must be isolated. Anyway, if a Sybil attacker enters the network, any of its neighbours will detect it as soon as it sends proofs of life for different vertexes ID.

4 Performance Analysis

We now analyze the efficiency of the proposal both from the energy consumption and from computational complexity points of view. We consider the energy consumption which is the result of transmissions of data and processor activities due to authentication tasks. In the proposal there are two phases when computational overhead is more significant: the ZKP-based access control and the periodic checking of stored FIFO queue. A reduction on the number of rounds of ZKP has a direct effect on the total exchanged messages size in insertions, but a trade-off should be maintained between protocols robustness and performance. Indeed, regarding total data transmission over wireless links, the ZKPs take less than 10% in a usual situation.

The periodic proofs of life accounts for approximately 90% of the total exchanged message size in many cases. However, we have found that these compulsory proofs of life imply an incentive technique for stimulating cooperation



```
Node: 13: has been succesful with its broadcast, so HC is now: 8 3 9 7 4 14 2 13 6 1 10 0
HC and graph have changed due to the insertion of 2 HC edges and 2 non-HC edges to the graph
HC: 8 3 9 7 4 14 2 13 6 1 10 0 is sent together with information about the last period of (n^3) /4
```

Fig. 3. Example of Network Simulation with NS-2

Table 1. Example of Trace

Time	Event	H.C.
0.1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 are legitimate	8,3,9,7,4,2,6,5,1,10,0
1.29	Insertion of Node 14 is broadcast by Node 4	8,3,9,7,4,14,2,6,5,1,10,0
1.30	Nodes 3, 1, 0 do not answer to the proof of life	
3.29	Node 0 reaches 8 and starts a ZKP for re-insertion	
8.69	Node 3 reaches 4 and starts a ZKP for re-insertion	
9.40	Node 1 reaches 10 and starts a ZKP for re-insertion	
11.65	Node 1 turns off	
13.97	Proof of life started by Node 3	
14.27	Nodes 1, 2 do not answer to the proof of life	
14.82	Node 2 reaches 14 and starts a ZKP for re-insertion	
17.27	Proof of life started by Node 2	
17.57	Nodes 1, 5 do not answer to the proof of life	
21.71	Node 5 turns off	
31.40	Node 1 turns on and Node 2 is chosen for the ZKP	
31.46	Node 4 turns off	
32.51	Proof of life started by Node 1	
32.78	Nodes 4, 5, 6 do not answer to the proof of life	
34.29	Node 6 reaches 2 and starts a ZKP for re-insertion	
38.51	Proof of life started by Node 6	
38.79	Nodes 4, 5 do not answer to the proof of life	
41.46	Node 1 turns off	
53.25	Node 1 turns on and Node 0 is chosen for the ZKP	
59.61	Proof of life started by Node 6	
59.99	Nodes 4, 5 do not answer to the proof of life	
64.26	Node 5 is deleted	8,3,9,7,4,14,2,6,1,10,0
64.71	Node 2 turns off	
72.58	Node 4 turns on and Node 0 is chosen for the ZKP	
75.41	Insertion of Node 13 is broadcast by Node 14	8,3,9,7,4,14,2,13,6,1,10,0
75.43	Node 2 does not answer to the proof of life	

in authentication tasks. This is due to the fact that nodes that are broadcasters of deletions or authenticators in insertions or access controls are exempted from their obligation to broadcast their proofs of life.

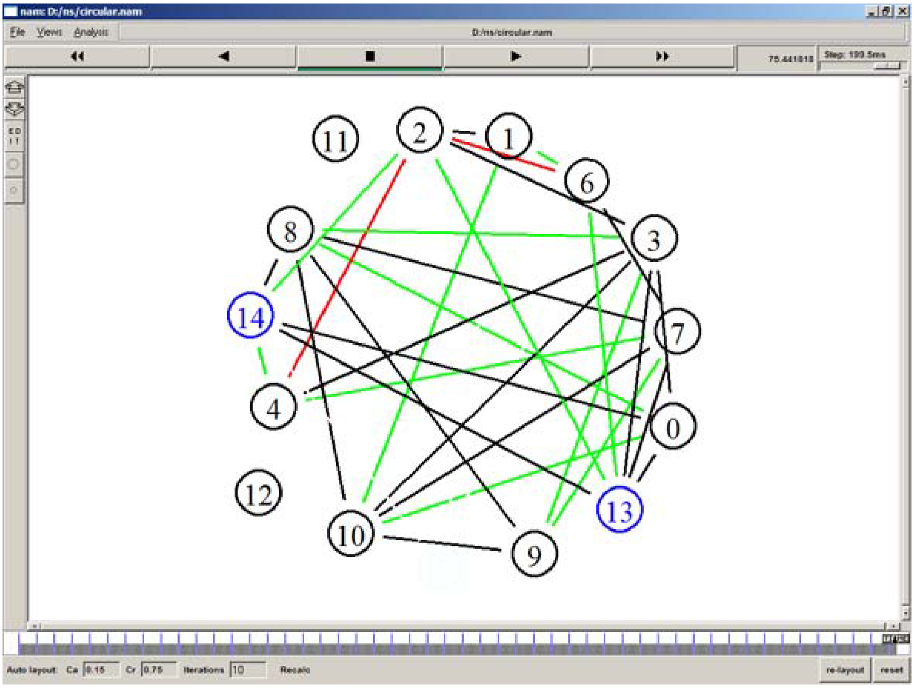
In order to reduce data communication cost of the protocol, an increase on the threshold period T might be an option, but again an acceptable balance should be kept. According to our experiments, T should depend directly on the number of legitimate and/or on-line nodes in order to prevent a possible bandwidth overhead of large networks.

For the performance analysis of the proposal we used the Network Simulator NS-2 with DSR routing protocol. We created several Tcl based NS-2 scripts in order to produce various output trace files that have been used both to do data processing and to visualize the simulation. Within our simulation we used the visualization tool of Network Animator NAM and the NS-2 trace files analyzer of Tracegraph. For the simulation of mobility we used the setdest program in order to generate movement pattern files using the random waypoint algorithm.

An example of simulation is shown graphically in Figure 3. Basically it consists of generating a scenario file that describes the movement pattern of the nodes and a communication file that describes the traffic in the network. These files are used to produce trace files that are analyzed to measure various parameters. An excerpt of the trace files corresponding to the same example is shown in Table 1.

The trace files are used to visualize the simulation using NAM, while the measurement values are used as data for plots with Tracegraph. The final graph and hamiltonian cycle associated to the example network is shown in Figure 4 where green is used to indicate the hamiltonian cycle, blue is used for the inserted nodes and red is used for the edges deleted from the hamiltonian cycle when inserting new nodes.

We conducted many different simulations in order to see the effects of different metrics by varying network density and topology. In particular, we varied the number of nodes from 15 to 100, the area from 400x400 to 800x800 m^2, and the period of simulation from 60 to 200 seconds. We also changed the probabilities of insertions and deletions in each second from 5% to 25%, in order to modify the mobility rate and antenna range of nodes from 2 to 15 m/s and 100 to 250



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

Fig. 4. Example of Final Associated Graph and Hamiltonian Cycle

meters respectively. This range also defines different frequencies of accesses to the network.

The first conclusions that we have obtained from the simulations are:

- The protocol scales perfectly to any sort of networks with different levels of topology changes.
- Node density is a key factor for the mean time of insertions, but such a factor is not as big as it might be previously assumed since nodes do not forward two packets of data corresponding to the same proof of life coming from two different nodes.
- A right choice of parameter T should be done according to number of nodes, bandwidth of wireless connections and computation and storing capacities of nodes.
- A positive aspect of the proposal is that the requirements in the devices' hardware are very low.

5 Conclusions and Open Questions

This work describes a new authentication scheme that has been specially designed for MANETs. Such a protocol supports knowledge-based member authentication in server-less environments. The overall goal of this proposal has been to design a strong authentication scheme that is able to react and adapt to network topology changes without the necessity of any centralized authority. Its core technique consists of a Zero-Knowledge Proof, in order to avoid the transference of any relevant information. Furthermore, the proposal is balanced since the procedures that the legitimate members of the network have to carry out when the network is upgraded (insertion or deletion of nodes) imply identical work for every legitimate member of the network.

The development of an initial simulation of the proposal through the NS-2 network simulator has been carried out. The definitive simulation results will be included in a forthcoming version of this work. Also, the study of different applications, practical limitations and possible extensions of the proposed scheme may be considered open problems.

References

1. Aboudagga, N., Tamer, M., Eltoweissy, M., DaSilva, L., Quisquater, J.J.: Authentication protocols for ad hoc networks: Taxonomy and research issues. In: Proceedings of the 1st ACM international workshop on Quality of service and security in wireless and mobile networks (October 2005)
2. Asaeda, H., Rahman, M., Manshaei, H., Fukuzawa, Y.: Implementation of Group Member Authentication Protocol in Mobile Ad-hoc Networks. In: Proceedings of IEEE Wireless Communications and Networking Conference WCNC, Las Vegas, USA (April 2006)
3. Caballero-Gil, P., Hernández-Goya, C.: Strong solutions to the identification problem. In: Wang, J. (ed.) COCOON 2001. LNCS, vol. 2108, pp. 257–261. Springer, Heidelberg (2001)

4. Caballero-Gil, P., Hernández-Goya, C.: Zero-Knowledge Hierarchical Authentication in MANETs. IEICE Transactions on Information and Systems E-89-D, 1288–1289 (2006)
5. Capkun, S., Buttyan, L., Hubaux, J.P.: Self-organized public-key management for mobile ad-hoc networks. IEEE Transactions on Mobile Computing (2003)
6. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (1987)
7. Hahm, S., Jung, Y., Yi, S., Song, Y., Chong, I., Lim, K.: A Self-Organized Architecture in Mobile Ad-Hoc Networks. In: Kim, C. (ed.) ICOIN 2005. LNCS, vol. 3391, pp. 689–696. Springer, Heidelberg (2005)
8. Hubaux, J.P., Buttyán, L., Capkun, S.: The quest for security in mobile ad hoc networks. In: Proceedings of MobiHoc, pp. 146–155 (2001)
9. Jiang, T., Baras, J.S.: Graph Algebraic Interpretation of Trust Establishment in Autonomic Networks. Wiley Journal of Networks Submitted (under review)
10. Maki, S., Aura, T., Hietalathi, M.: Robust membership management for ad-hoc groups. In: Proceedings of 5th Nordic Workshop on Secure IT Systems NORDSEC (2000)
11. Saxena, N., Tsudik, G., Yi, J.H.: Efficient node admission for short-lived mobile ad hoc networks. In: IEEE International Conference on Network Protocols ICNP, pp. 269–278 (November 2005)
12. Weimerskirch, A.: Authentication in Ad-hoc and Sensor Networks. Ph.D. Thesis Ruhr-University Bochum. Germany, (July 2004)
13. Wierzbicki, A., Zwierko, A., Kotulski, Z.: A New Authentication Protocol for Revocable Anonymity in Ad-Hoc Networks. In: Proceedings of the IASTED Communication, Network, and Information Security CNIS, Phoenix, AZ, USA (2005)
14. Wu, H.C., Hwang, M.S., Liu, C.H.: A Secure Strong-Password Authentication Protocol. Fundamenta Informaticae 68, 399–406 (2005)
15. Zhou, L., Haas, Z.: Securing ad hoc networks. IEEE Networks 13, 24–30 (1999)

An Efficient Pre-authentication Scheme for IEEE 802.11-Based Vehicular Networks

Junbeom Hur, Chanil Park, and Hyunsoo Yoon

Korea Advanced Institute of Science and Technology(KAIST)
{jbhur,chanil,hyoon}@nslab.kaist.ac.kr

Abstract. In vehicular networks, supporting user mobility is one of the most challenging issues. Recently, as the desires for the high mobility and high-quality real-time services increase, fast handoff among base stations comes to a center of quality of connections. Therefore, minimizing re-authentication latency during handoff is crucial for revolutionizing the driving environment on public vehicular networks in terms of safety and convenience. In this study, we propose an efficient pre-authentication scheme for fast and secure handoff in IEEE 802.11-based vehicular networks. The proposed scheme reduces the handoff delay by reducing 4-way handshake to 2-way handshake between an access point and a mobile vehicle station during the re-authentication phase. Furthermore, the proposed scheme gives little burden over the proactive key pre-distribution scheme while satisfying 802.11i security requirements.

Keywords: proactive key distribution, fast handoff, pre-authentication, IEEE 802.11 network, vehicular network.

1 Introduction

The vehicular network has the potential to significantly enhance the driving environment providing increased safety and convenience. For many applications, vehicular networks will require real-time, or near real-time, responses as well as hard real-time guarantees. Differing from traditional sensor networks, or ad hoc networks, vehicular networks typically require faster responses [1]. From these perspectives, for instance, a real-time emergency application notifying vehicles of a sudden deceleration or accidents neighboring them would prevent potential accidents entirely. This network environment motivates the need for an infrastructure that will provide drivers with access to a variety of vital vehicular and roadside information.

Nowadays, the dedicated short range communications (DSRC) [2], based on an extension to the IEEE 802.11 standard [4] is emerging as a competent technology to meet the requirements of users for disseminating traffic information and accessing to roadside base stations for quality services [3]. Due to the lack of mobility support of IEEE 802.11, however, seamless mobile services, particularly for real-time applications are hard to be served in IEEE 802.11-based vehicular networks when a vehicle station (STA) moves from one access point

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 121–136, 2007.
© Springer-Verlag Berlin Heidelberg 2007

(AP) to another on the roadside. Especially, Authentication, Authorizing, and Accounting (AAA) servers are supposed to be located far away from each AP so that the full authentication delay requires about $1000ms$ [16]. This excessive latency of complete user authentication and security negotiations which should be performed at each AP during handoff can be a main obstacle to seamless services for real-time applications in a high mobile vehicular network. Therefore, fast re-authentication and re-association schemes are essential during handoff between APs.

A logical handoff procedure in IEEE 802.11-based wireless networks consists of three phases: detection, search, and execution. In the detection phase, the STA discovers that it is out of range of its current AP. At this point, the STA launches the search phase where the STA seeks to identify a candidate set of next APs via active or passive means [4]. Once the candidate set of next APs has been identified, the STA selects the next AP and begins re-association and re-authentication with it. The current IEEE 802.11i [5] security architecture recommends an authentication process to follow EAP/TLS [8]. In addition, IEEE 802.11i makes use of IEEE 802.1x [7] model to authenticate the STA to the AAA server using AAA protocols like a Remote Authentication Dial-In User Service (RADIUS) [9] to prohibit unauthorized access to the network. The complete EAP/TLS authentication, however, causes too large latency to support multimedia services whose overall latency should not exceed about $50ms$ [15].

To solve this problem, many previous studies proposed fast handoff schemes in diverse aspects. A. Mishra et al. proposed the proactive key distribution (PKD) scheme using the mobility topology of the network, Neighbor Graph, which tracks the potential APs to which a STA may handoff in the near future [10]. This scheme reduces the handoff delay by pre-authenticating the STA to the next neighbor APs before handoff. To predict more precise neighbor graph considering handoff pattern of the STA, S. Pack et al. gathered statistics on the mobility pattern of a STA and introduced the frequent handoff region (FHR) which have the highest probabilities to be the next AP in the selective neighbor caching (SNC) scheme [12]. To reduce the authentication exchange duration, M. Kassab et al. proposed two pre-authentication schemes: PKD with anticipated 4-way handshake, and PKD with inter AP protocol (IAPP) caching [13]. However, these schemes heavily burden the current AP with additional communication and computation efforts in that the current AP should pre-establish independent private session keys for its neighbor APs and transmit the keys to all of its neighbor APs. To make matters worse, the security cannot be guaranteed because APs are supposed to know private keys of their neighbor APs.

In this study, an efficient pre-authentication scheme enhancing the proactive key distribution method is proposed. The proposed scheme reduces the number of exchanges for private session key generation between an STA and an AP of the re-association phase by exchanging key-generating materials in the pre-authentication phase before handoff. Therefore, the re-authentication delay of 4-way handshake during handoff can be reduced to that of 2-way handshake. In addition, the proposed scheme guarantees security requirements of IEEE 802.11i

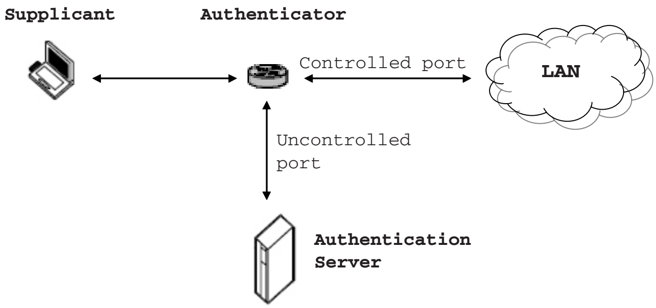


Fig. 1. IEEE 802.1x architecture

standard and secure communications between an STA and each AP. The proposed scheme can be adapted to a typical IEEE 802.11-based local area network (LAN) where the requirement for real-time responses is crucial as well as the high mobile vehicular networks.

The paper is organized as follows. In Section 2, we describe IEEE 802.11i-based handoff mechanisms using the PKD method. In Section 3, we propose an efficient pre-authentication scheme enhanced from the PKD method. In Section 4, we evaluate the performance of the proposed scheme compared with the PKD-based pre-authentication methods and make a security analysis. In Section 5, we show a simulation result, and remark the conclusion of the paper in Section 6.

2 Related Work

In this section, a brief description on the handoff mechanism in IEEE 802.11 wireless networks is introduced. IEEE 802.11i uses IEEE 802.1x [7] framework to authenticate and authorize devices connected to the network at the link layer for various link technologies. The IEEE 802.1x has three main entities: the supplicant, authenticator, and authentication server (AS). In a vehicular network scenario using the IEEE 802.11i, the roadside access point AP, vehicle station STA, and AAA server take the role of the authenticator, supplicant, and AS of the IEEE 802.1x architecture, respectively. In the architectural framework of the IEEE 802.1x standard as described in Fig. 1, a supplicant who desires to use a service authenticates to the central AS via the uncontrolled port on the authenticator. The authentication process between the AS and the supplicant via the authenticator is carried over an extensible authentication protocol (EAP) [8] like an EAP/TLS. Once the AS and the supplicant mutually authenticated each other, the authenticator and the supplicant establish keying materials through an EAPOL-key exchange [6]. After that, the AS directs the authenticator to allow the STA to access the network through the controlled port on the authenticator. Typically, the AS and authenticator communicate with each other using RADIUS [9] protocol.

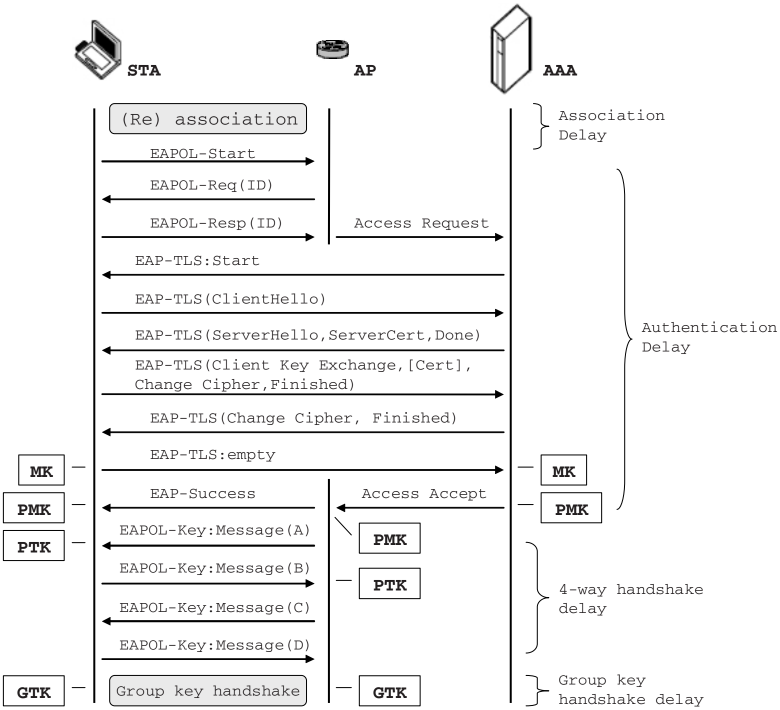


Fig. 2. Complete EAP/TLS authentication exchange

2.1 EAP/TLS Authentication

In the IEEE 802.11i authentication process using the EAP/TLS, the STA and the AAA server mutually authenticate each other based on a certificate from a common trusted certificate authority (CA). The mutual authentication process drives the STA and the AAA server to share a strong secret master key (MK) and to initialize one-way pseudo-random functions (PRF) for generating further key materials. The STA and the AAA server generate a pairwise master key (PMK) separately using the following equation [1]:

$$PMK = PRF(MK, \text{`}client\ EAP\ encryption\text{'} \mid ClientHello.random \mid ServerHello.random). \quad (1)$$

The AAA server then sends the PMK to the associated AP. After that, the STA and the associated AP perform 4-way handshake through the EAPOL protocol [6] to confirm the PMK between them and to derive a session key, pairwise transient key (PTK). The 4-way handshake is described as follows:

1. *Message(A): EAPOL-Key(ANonce, Unicast)*
   - This message contains ANonce, which is a nonce value generated by the AP. Once the STA has received this message, the STA can derive a PTK. This message is not encrypted or integrity-verified.
2. *Message(B): EAPOL-Key(SNonce, Unicast, MIC)*
   - This message contains SNonce, which is a nonce value generated by the STA, and a message integrity check (MIC) to protect its integrity. The AP derives the PTK using SNonce and verifies the MIC. If this step succeeds, the AP can confirm that the STA has the correct PMK and PTK, and that there is no man-in-the-middle attack.
3. *Message(C): EAPOL-Key(Install PTK, Unicast, MIC)*
   - This message tells the STA that the AP is ready to begin encryption using PTK. If this step succeeds, the STA can verify that the AP has the correct PMK and PTK, and that there is no man-in-the-middle attack.
4. *Message(D): EAPOL-Key(Unicast, MIC)*
   - After this message is sent, both sides install the PTK and begin data encryption using the PTK.

During the 4-way handshake, the STA and the AP generate PTK separately using following equation (2):

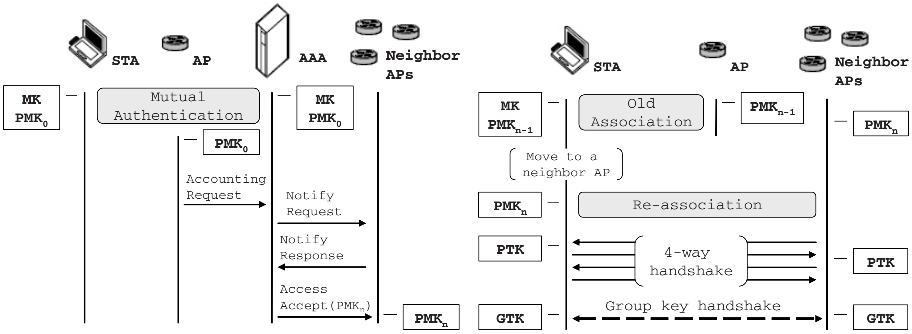
$$PTK = PRF(PMK, ANonce, SNonce, STA_{mac}, AP_{mac}), \qquad (2)$$

where STA_{mac} and AP_{mac} represent the MAC addresses of the STA and the AP, respectively. The PTK is shared only between the STA and the currently associated AP for secure communication between them. The confidentiality of the PTK is only based on a secrecy of the PMK because other key-generating materials are exposed. Fig. 2 describes the complete message exchanges and the point of each key generation time during a complete EAP/TLS authentication.

The latency of the full EAP/TLS authentication exchanges is estimated to about $1000ms$, which takes an excessive delay compared with the $50ms$ for real-time applications [16]. Thus, to support a fast authentication during handoff in IEEE 802.11-based networks, the proactive key distribution scheme using neighbor graphs of APs was proposed recently [10].

2.2 Proactive Key Distribution

The proactive key distribution (PKD) scheme [10] pre-authenticates an STA to next APs by pre-distributing authentication keys, PMKs, to the neighbor APs of the currently associated AP before handoff. Thus, upon handoff, re-authentication exchange between the STA and the AP is reduced to the 4-way handshake and the group key handshake. The neighbor graph for all APs in the network is managed by the AAA server en masse. To predict and make a precise neighbor graph can be one of the challenging issues in the wireless handoff, however, it is out of our interests in this paper.



(a) Pre-authentication

(b) Re-authentication

Fig. 3. Authentication exchange process with PKD

In the PKD method, the PMK is generated through following equations (3):

$$PMK_0 = PRF(MK, \text{'}client\ EAP\ encryption\text{'} \mid ClientHello.random \mid ServerHello.random),$$

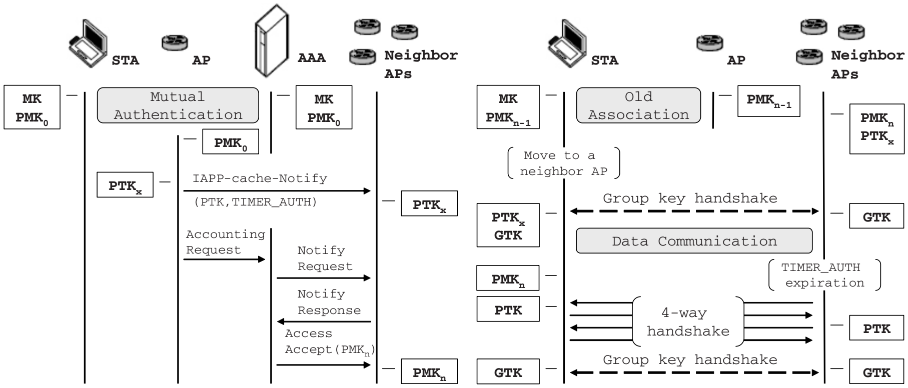
$$PMK_n = PRF(MK, PMK_{n-1} \mid AP_{mac} \mid STA_{mac}), \tag{3}$$

where n represents the n^{th} re-association. The PMK_0 is generated during a first mutual authentication between an STA and an AAA server. The AAA server pre-distributes the PMK_n to next neighbor APs for pre-authentication. This prevents other dissociated APs from generating the PTK of the currently associated AP and the STA, which follows the IEEE Task Group I (TGi) trust assumption that the only associated AP and the AAA server are trusted [10].

The pre-authentication processes with PKD method are described in Fig. 3(a). After a first authentication, the current AP requests the AAA server to pre-authenticate the associated STA to its neighbor APs for a predictive handoff via Accounting-Request message. Upon receiving it, the AAA server multicasts Notify-Request messages to the neighbor APs, and the neighbor APs respond to the AAA server with Notify-Response messages. Then, the AAA server generates PMK_n using equation (3) and sends it to the neighbor APs.

Next, the re-authentication processes are described as follows. As soon as the STA moves to a next AP, the STA gets the MAC addresses of the AP, generates a new PMK_n, and compares it with the PMK_n of the AP which was given by the AAA server in the pre-authentication phase.

Thus, the mutual authentication process between an STA and an AAA server after handoff is reduced to perform 4-way handshake and group key handshake as shown in Fig. 3(b). The PKD method reduces the full authentication delay of about 1000ms to the re-authentication delay of 60ms [16], but it still exceeds the expected latency for real-time applications in vehicular networks.



(a) Pre-authentication (b) Re-authentication

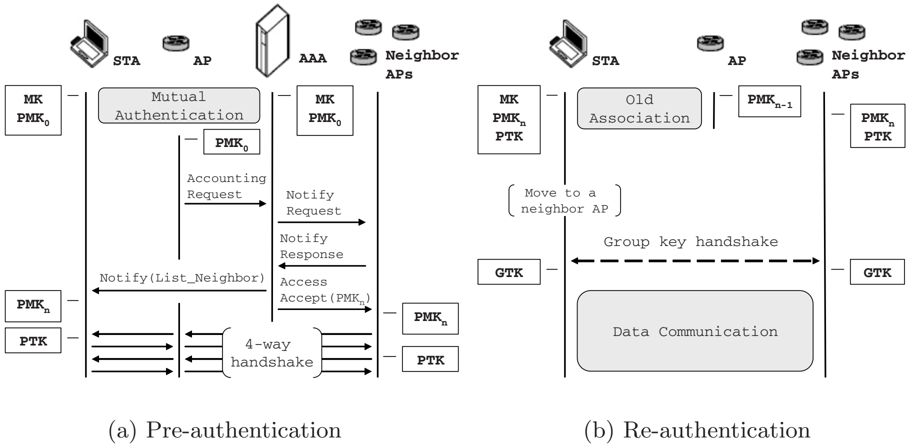
Fig. 4. Authentication exchange process with ‘PKD with IAPP Caching’

2.3 Other Approaches for Pre-authentication

Recently, M. Kassab et al. proposed two pre-authentication methods based on the PKD method [13]. The main idea of these methods is to reduce the re-authentication delay by performing 4-way handshake in the pre-authentication phase at the expense of additional loads at the AP and the STA, and security degradation.

PKD with IAPP Caching. In the PKD with IAPP caching method, a current AP calculates all the PTK_x for its neighbor AP_x separately using the PMK, and pre-distributes the PTK_x and its valid time value to the corresponding neighbor AP_x through the inter access point protocol (IAPP) [6] as described in Fig. 4(a). Upon handoff to a new AP_x, the STA derives the PTK_x and authenticates itself to the AP with the PTK_x through the group key handshake. Thus, the re-authentication phase is reduced to the group key handshake process without 4-way handshake as shown in Fig. 4(b). This re-authentication, however, is temporary authentication, which remains valid only within the time limit. After the time limit, the STA and the AP should authenticate each other and generate a permanent PTK for secure channel again.

PKD with Anticipated 4-way Handshake. In the PKD with anticipated 4-way handshake method, an STA and neighbor APs perform 4-way handshake through the current AP in the pre-authentication phase in advance. Thus, this method also reduces the re-authentication delay to the only group key handshake delay. To carry out 4-way handshake, the STA receives a list and MAC addresses of neighbor APs of the current AP from the AAA server. So, the STA can generate PTKs with the neighbor APs through its current AP using PMK_ns. Each authentication exchange process is described in Fig. 5.



(a) Pre-authentication

(b) Re-authentication

Fig. 5. Authentication exchange process with 'PKD with Anticipated 4-way Handshake'

3 Efficient Proactive Key Distribution

In this section, a pre-authentication scheme based on the PKD method is proposed for fast handoff in the IEEE 802.11-based vehicular network environment. The main idea of the proposed scheme is to perform 2-way handshake during a pre-authentication phase and perform remaining 2-way handshake during a re-authentication phase while satisfying security requirements of the IEEE 802.11i standard.

3.1 Modified EAP/TLS Authentication

To exchange the nonce values between the STA and its neighbor APs in pre-authentication phase, the full EAP/TLS authentication exchange should be little modified as described in Figure 6. Note that the STA and AP exchange their nonce values before 4-way handshake in the modified EAP/TLS authentication.

The STA transmits its nonce values to an AAA server through the following modified message exchange during the first full EAP/TLS authentication:

$$\textit{EAP-TLS:empty} \longrightarrow \textit{EAP-TLS(SNonce)}.$$

Then, the AAA server stores the nonce value received from the STA and delivers it with the PMK to the associated AP. Upon receiving the nonce value, the AP can generate the PTK for the STA. Then, the AP transmits its nonce value and MIC to the STA to verify that the AP has the correct PMK and PTK through the modified EAP-Success message exchange:

$$\textit{EAP-Success} \longrightarrow \textit{EAP-Success(ANonce, MIC)}.$$

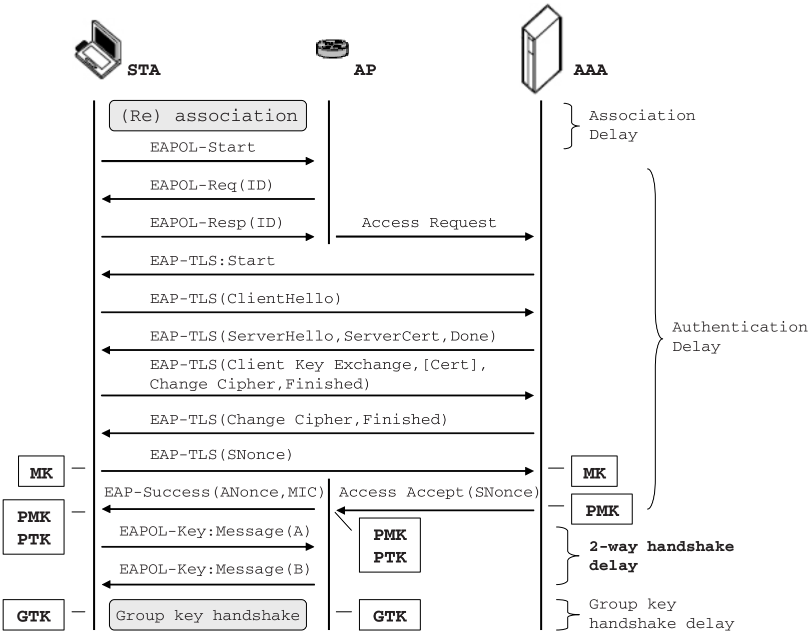


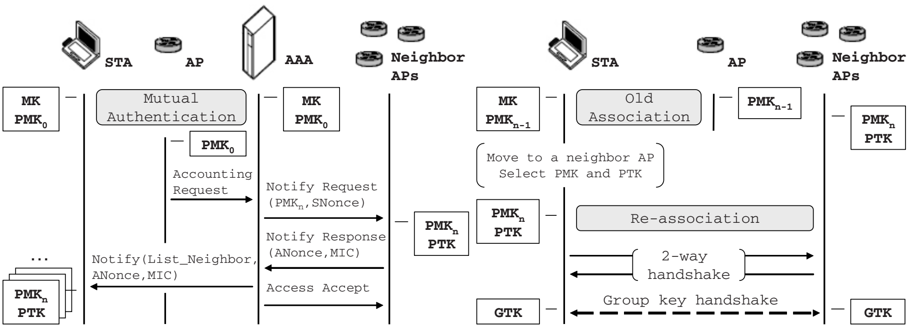
Fig. 6. Modified EAP/TLS authentication exchange

Therefore, thereafter only 2-way handshake is required to establish the PTK between the STA and the AP, and check the integrity of the keying materials. The 2-way handshake process is described as follows:

1. *Message(A): EAPOL-Key(Install PTK, Unicast, MIC)*
   - This message tells the AP that the STA is ready to begin encryption using the PTK. If this step succeeds, the AP can verify that the STA has the correct PMK and PTK, and that there is no man-in-the-middle attack.
2. *Message(B): EAPOL-Key(Unicast, MIC)*
   - After this message is sent, both sides install the PTK and begin data encryption using the PTK.

3.2 Authentication with the Efficient PKD

After the first mutual authentication between an STA and an AAA server, the AAA server requests neighbor APs of the current AP to pre-authenticate the STA by sending the corresponding PMK and SNonce of the STA. Upon receiving them from the AAA server, the neighbor APs generate their own PTK for the STA during a pre-authentication phase and respond to the AAA server with their own nonce values and MICs of the message. Upon receiving them, the AAA server transmits a list of neighbor APs, their nonce values, and MICs to



(a) Pre-authentication

(b) Re-authentication

Fig. 7. Authentication exchange process with efficient PKD

the STA. After that, the STA generates PMKs and PTKs corresponding to each neighbor AP and verifies that each neighbor AP has the correct PMK and PTK. If these steps succeed, the AAA server completes the pre-authentication phase by transmitting an access accept message to the neighbor APs as described in Fig. 7(a).

Upon handoff, the STA selects the corresponding PMK and PTK to the re-associated AP among the keys which were generated in the pre-authentication phase. Then, the STA and the AP check for the integrity of the keys and install the PTK by performing 2-way handshake. Therefore, re-authentication exchange between the STA and the AP is reduced to the 2-way handshake and the group key handshake as in Fig. 7(b). Compared to the PKD method, the proposed scheme requires one more additional communication exchange for the list of neighbor APs in the pre-authentication phase while reducing 4-way handshake to 2-way handshake in the re-authentication phase.

4 Protocol Analysis

4.1 Performance Analysis

In this section, we analyze and compare the performance of the authentication schemes: PKD, PKD with IAPP caching, PKD with anticipated 4-way handshake, and the proposed scheme. The overall results of the analysis are summarized in Table 1 in which m denotes the average number of neighbor APs per each AP.

In Table 1, the communication factor represents the necessary number of message exchanges for the PMK and PTK establishment among the entities. The common exchanges of the first full EAP/TLS authentication exchanges, or group key handshake are not included in this analysis. The computation factor represents secret keys, which should be generated by each entity per handoff except

Table 1. Performance analysis of authentication schemes

		PKD	PKD with IAPP caching	PKD with anticipated 4-way handshake	Proposed scheme
Communication	Pre-auth.	m (PMK)	m (PMK), m (PTK)	m (PMK) + 1 (list), $2m\times$4-way handshake	m (PMK) + 1 (list)
	Re-auth.	4-way handshake	<permanent> 4-way handshake, group key handshake	0	2-way handshake
Computation	STA	PMK_n, PTK, GTK	<temporary> PTK, GTK, <permanent> PMK_n, PTK, GTK	PMK_n, GTK, $m\times$ PTK	PMK_n, PTK, GTK
	AP	0	$m\times$ PTK	0	0
Memory Requirement	STA	0	0	local NG	local NG
	AP	0	local NG	0	0
IEEE 802.11i Security		Y	N	Y	Y

the common key PMK_0 and MK. The memory requirement factor represents the memory consumption for a neighbor graph (NG) of the current AP, which should be maintained by each entity for key generation in the pre-authentication phase. The AP in Table 1 indicates the current AP.

The IEEE 802.11i security factor represents whether the schemes satisfy the security requirements of the IEEE 802.11i standard: (1) There should be mutual authentication and fresh key derivation at each AP, (2) Mutual authentication should not cause man-in-the-middle attack. The PKD with IAPP caching method is very vulnerable to the AP's compromise and the man-in-the-middle attack because each AP should participate in the process of secret key establishment for other APs. Thus, even a single AP's compromise can be a great threat to the security of the whole network.

The total communication exchanges for authentication of the proposed scheme is the least compared to the other schemes. The PKD with anticipated 4-way handshake scheme has the shortest re-authentication delay; however, as the network size and the velocity of mobile vehicle STAs increase, the total authentication efforts of the scheme may increase most greatly due to the overburdened pre-authentication process.

Compared to the PKD method, the proposed scheme requires one more communication exchange in the pre-authentication phase and additional storage from the STA for a neighbor list, but reduces the 4-way handshake to the 2-way handshake while keeping the other protocol exchanges intact and satisfying IEEE 802.11i security requirements. This can make the secure and seamless multimedia services in IEEE 802.11-based vehicular network to be practical whose handoff latency should be less than $50ms$ in that the re-authentication delay would be reduced from $60ms$ to the half.

4.2 Security Analysis

Key Freshness. To guarantee the freshness of a key derived at each AP, how to refresh the nonce value of an STA can be one of the considerable issues in the proposed scheme. Although the freshness of the PTK can be guaranteed by the freshness of the ANonce, a reuse of the SNonce may make a system vulnerable to the replay attack. An attacker who masqueraded as a participant in the system by forging a MAC address can eavesdrop on every message, remember nonces and MICs of each message, insert forged messages, and replay stored messages with a combination of known nonces and MICs. To refresh the nonce value of the STA, it can be one solution for a trusted AAA server to regenerate the SNonce on behalf of the STA and distribute it to neighbor APs like the PMK_n pre-distribution. That is,

$$SNonce_n = PRF(MK, SNonce_{n-1}, STA_{mac}, AP_{mac}),$$

where n represents n^{th} re-association of the STA. This nonce value generation process can achieve the freshness of the PTK. In addition, because the MK is securely shared between the STA and the AAA server, no other participants but they can generate or predict the appropriate SNonce per handoff. Under the assumption that the AAA server is totally trusted and cannot be exploited by an outsider adversary, the randomness of the SNonce can be guaranteed to the adversary since an adversary cannot guess the secret MK. Thus, the key refreshness is guaranteed.

DoS Attack. According to the security verification of 4-way handshake using Murφ model in [17], the 4-way handshake is analyzed to be vulnerable to a simple attack on $Message(A)$ that causes PTK inconsistency between the AP and the STA. The attacker who is impersonating the authenticator sends a forged $Message(A)$ to the STA after $Message(B)$ of the 4-way handshake. The STA will then calculate a new PTK corresponding to the nonce for the newly received $Message(A)$, leading to PTK inconsistency so that the subsequent handshakes to be blocked. The vulnerability of the 4-way handshake to DoS attack on the $Message(A)$ is actualized by the AP-initiated 4-way handshake in which the STA should must accept all messages to allow the handshake to proceed while the AP can initiate only one handshake instance and accept only the expected response within the expected time. So, the memory exhaustion attack on the STA always exists.

In the proposed scheme, however, the STA initiates the handshake, thus the STA needs not store all the unexpectedly received nonces and derived PTKs. This prevents the memory exhaustion attack on the typically resource-constrained STA. However, the STA-initiated 4-way handshake is still vulnerable to the DoS attack on the $Message(A)$. One possible solution is to add a MIC to the $Message(A)$ using a common secret such as a PMK to prevent an attacker from forging it, and to use a sequence counter to defend against a replay attack.

4.3 Correctness Analysis

Here, we analyze our pre/re-authentication scheme using a logic-based formal analysis tool [18],[19] to ensure that our authentication protocol functions correctly. We deem that authentication is complete between the STA and AP if there is a PTK such that both believe($\models$) the share of it($\overset{PTK}{\leftrightarrow}$):

$$STA \models STA \overset{PTK}{\leftrightarrow} AP, \qquad AP \models STA \overset{PTK}{\leftrightarrow} AP.$$

We idealize the protocol below, with STA and AP_n as the principals, AS as the AAA server, N_s and N_a as the nonce values, and $MIC_k\{m\}$ as the MIC of the message m encrypted under the key k.

(Pre-authentication)
$Message$ 1. $AS \rightarrow AP_n : PMK,\ N_s$
$Message$ 2. $AP_n \rightarrow AS : N_a,\ MIC_{PTK}\{N_a\}$
$Message$ 3. $AS \rightarrow STA : MAC_{AP},\ N_a,\ MIC_{PTK}\{N_a\}$
(Re-authentication)
$Message$ 4. $STA \rightarrow AP_n : MIC_{PTK}\{\}$
$Message$ 5. $AP_n \rightarrow STA : MIC_{PTK}\{\}$

To analyze this protocol, we first give the following assumptions:

$$\begin{array}{ll}
STA \models STA \overset{PMK}{\leftrightarrow} AS, & AS \models STA \overset{PMK}{\leftrightarrow} AS, \\
AS \models STA \overset{PMK}{\leftrightarrow} AP_n, & STA \models AS| \sim N_s(\text{AS conveyed } N_s), \\
AP_n \models AS \triangleleft N_s(\text{AS is told } N_s), & STA \models \sharp(N_a)(N_a \text{ is fresh}), \\
AP_n \models \sharp(N_s). &
\end{array}$$

We analyze the idealized version of our authentication protocol by applying logical postulates of [18] and [19] to the assumptions; the analysis is straightforward. For brevity, we do not describe our deductions, and simply list the final results:

Analysis of $Message$ 1.	$AP_n \models STA \overset{PTK}{\leftrightarrow} AP_n$
Analysis of $Message$ 2,3.	$STA \models STA \overset{PTK}{\leftrightarrow} AP_n$
Analysis of $Message$ 4.	$AP_n \models STA \models STA \overset{PTK}{\leftrightarrow} AP_n$
Analysis of $Message$ 5.	$STA \models AP_n \models STA \overset{PTK}{\leftrightarrow} AP_n$

This state achieves more than the complete condition of the authentication. Each principal, STA and neighbor APs, knows a shared secret, PTK, with each other and has a knowledge of a shared secret that he believes the other will accept as being shared by the two principals. From this point, they can transfer data securely.

5 Simulation

In this section, the maximum velocity of a vehicle that each authentication scheme can support is analyzed and the simulation result is shown. In the simulation, it is assumed that the pre-authentication of a vehicle is performed to m one-hop neighbor APs. Let $\varDelta T_x$ and RTT_{AAA} be the time duration needed to perform a process x and round-trip time to AS, respectively. Let $\varDelta T_{IAPP}$ and $\varDelta T_{NoIAPP}$ be the association latency using the IAPP protocol and using no IAPP protocol, respectively. Let D denote the radius of cell coverage of an AP, and C denote the length of the intersected cell coverage between two neighboring APs. For simplicity, a vehicle is assumed to go through the center of service cells of APs. Then, the maximum velocity V of a vehicle in each scheme is calculated using the pre-authentication latency ($\varDelta T_{preauth}$) and re-authentication latency ($\varDelta T_{reauth}$) in Table 2 as $V = MIN((D-2C)/\varDelta T_{preaut}, C/\varDelta T_{reauth})$, where MIN is represented as a minimum function. V is the maximum velocity that guarantees the complete key exchanges needed for fast handoff in the authentication schemes.

According to the latency budget estimated in [16], the simulation environments are set as follows: $\varDelta T_{scan} = 40ms$, $\varDelta T_{4way} = 60ms$, $\varDelta T_{reassoc} = 10ms$, $RTT_{AAA} = 50ms$, $\varDelta T_{IAPP} = 40ms$, $\varDelta T_{NoIAPP} = 2ms$. D is set to $30m$ and C is set to $5m$. Then the simulated maximum velocity of a vehicle in each scheme

Table 2. Authentication latency

	$\varDelta T_{preauth}$	$\varDelta T_{reauth}$
PKD	$(3m+1)/2 \times RTT_{AAA}$	$\varDelta T_{scan} + \varDelta T_{4way} + \varDelta T_{reassoc}$
PKD with IAPP caching	$m \times \varDelta T_{IAPP} +$ $(3m+1)/2 \times RTT_{AAA}$	<permanent> $\varDelta T_{scan} + \varDelta T_{4way} + \varDelta T_{reassoc}$ <temporary> $\varDelta T_{scan} + \varDelta T_{reassoc}$
PKD with anticipated 4-way handshake	$4m \times (\varDelta T_{IAPP} + \varDelta T_{NoIAPP})$ $+(3m+2)/2 \times RTT_{AAA}$	$\varDelta T_{scan} + \varDelta T_{reassoc}$
Proposed scheme	$(3m+2)/2 \times RTT_{AAA}$	$\varDelta T_{scan} + 1/2 \times \varDelta T_{4way} + \varDelta T_{reassoc}$

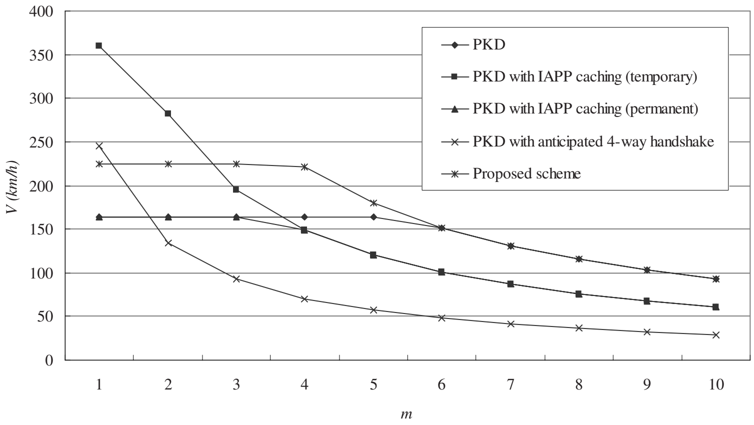


Fig. 8. Maximum velocity of a vehicle

can be shown like Fig. 8. The x-axis represents the number of m, and y-axis represents the maximum velocity that each scheme can support.

Fig. 8 shows that as m increases, the maximum velocity of a vehicle tends to be more affected by the pre-authentication latency than by the re-authentication latency. When $m = 1$, the PKD with anticipated 4-way handshake can support the highest velocity of a vehicle except the PKD with IAPP caching of temporary authentication (that is not a secure protocol). As m increases, however, the pre-authentication latency exceeds the re-authentication latency, thus the maximum velocity of a vehicle decreases drastically due to the pre-authentication process. The proposed scheme shows higher velocity than PKD when $m < 6$, but as m increases more than 6, V is also restricted by the pre-authentication latency, thus it can support almost the same velocity as PKD.

6 Conclusion

An efficient key distribution scheme for fast and secure handoff is an essential technology for secure and quality services in IEEE 802.11-based vehicular networks. In this study, an efficient pre-authentication scheme based on the PKD method is proposed. The proposed scheme clearly improves the PKD method by reducing the re-authentication delay to 2-way handshake by transmitting nonce values between the STA and APs in the pre-authentication phase without security degradation. Since the proposed pre-authentication scheme is simple and does not require any impractical trust relationship among network entities, it is expected to be extensively adapted to IEEE 802.11-based vehicular networks for fast and secure handoff.

Acknowledgements. This research was supported by the MOST(Ministry of Science and Technology)/KOSEF(Korea Science and Engineering Foundation) through the AITrc(Advanced Information Technology Research Center) and the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment). (IITA-2006-C1090-0603-0015)

References

1. Parno, B., Perrig, A.: Challenges in Securing Vehicular Networks. In: ACM Workshop on Hot Topics in Networks (HotNets-IV), College Park, Maryland (November 2005)
2. Standard specification for telecommunications and information exchange between roadside and vehicle systems - 5 GHz band dedicated short range communications (DSRC) Medium Access Control (MAC) and physical layer (PHY) specifications (September 2003)
3. Zarki, M.E., Mehrotra, S., Tsudik, G., Venkatasubramanian, N.: Security Issues in a Future Vehicular Network. European Wireless (February 2002)

4. IEEE 802.11: Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE (June 2003)
5. IEEE 802.11i: Amendment 6: Medium Access Control (MAC) Security Enhancements. IEEE Computer Society (July 2004)
6. IEEE 802.11f: Recommnded Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operatoin. IEEE (July 2003)
7. IEEE 802.1x: IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control. IEEE (June 2001)
8. Aboba, B., Simon, D.: PPP EAP TLS Authenticatoin Protocol. RFC 2716 (October 1999)
9. Rigney, C., Willats, W., Calhoun, P.: Remote Authentication Dial In User Service (RADIUS). RFC 2869, June 2000.
10. Mishra, A., Shin, M., Arbaugh, W.A.: Pro-active Key Distribution using Neighbor Graphs. IEEE Wireless Communications 11 (February 2004)
11. Pack, S., Jung, H., Kwon, T., Choi, Y.: A Selective Neighbor Caching Scheme for Fast Handoff in IEEE 802.11 Wireless Networks. ACM SIGMOBILE Mobile Computing and Communications Review (October 2005)
12. Pack, S., Choi, Y.: Fast Inter-AP Handoff Using Predictive Authentication Scheme in a Public Wireless LAN. IEEE Networks (August 2002)
13. Kassab, M., Belghith, A., Bonnin, J., Sassi, S.: Fast Pre-Authentication Based on Proacitve Key Distribution for 802.11 Infrastructure Networks. In: ACM Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP‘05) (13 October, 2005)
14. M. Shin, J. Ma, W.A. Arbaugh: The Design of Efficient Internetwork Authentication for Ubiquitous Wireless Communications. Technical Report CS-TR-4617, Digital Repository at the University of Maryland (January 2006)
15. International Telecommunication Union: General Characteristics of International Telephone Connections and International Telephone Circuits. ITU-TG.114 (1988)
16. B. Aboba: Fast Handoff Issues. IEEE 802.11-03/155r0, 2003.
17. He, C., Mitchell, J.C.: Analysis of the 802.11i 4-Way Handshake. In: ACM Workshop on Wireless Security (WiSe 2004) (October 2004)
18. Burrows, M., Abadi, M., Needham, R.: A Logic of Authentication. ACM Transactions on Computer Systems 8(1), 18–36 (1990)
19. Gong, L., Needham, R., Yahalom, R.: Reasoning about Belief in Cryptographic Protocols. In: IEEE Symposium on Security and Privacy, pp. 234–248. IEEE Computer Society, Silver Spring (1990)

Intrusion Detection and Identification System Using Data Mining and Forensic Techniques

Fang-Yie Leu, Kai-Wei Hu, and Fuu-Cheng Jiang

Department of Computer Science and Information Engineering, Tunghai University, Taiwan
leufy@thu.edu.tw

Abstract. Presently, most computers authenticate a user's ID and password before the user can log in. However, if the two items are known to hackers, there is a risk of security breach. In this paper, we propose a system, named the Intrusion Detection and Identification System (IDIS), which builds a profile for each user in an intranet to keep track of his/her usage habits as forensic features. In this way the IDIS can identify who the underlying user in the intranet is by comparing the user's current inputs with the features collected in the profiles established for all users. User habits are extracted from their usage histories by using data mining techniques. When an attack is discovered, the IDIS switches the user's inputs to a honey pot not only to isolate the user from the underlying system, but also to collect many more attack features by using the honey pot to enrich attack patterns which will improve performance of future detection. Our experimental results show that the recognition accuracy of students in the computer science department of our university is nearly 99.16% since they are sophisticated users. The recognition accuracy of those other than computer science students is 94.43%.

Keywords: Forensic Features, Intrusion Detection, Data Mining, Identifying Users.

1 Introduction

Being widely used and quickly developed in recent years, network technologies have provided us with new life and shopping experiences, particularly in the fields of e-business, e-learning and e-money. But along with network development, there has come a huge increase in network crime. It not only greatly affects our everyday life, which relies heavily on networks and Internet technologies, but also damages computer systems that serve our daily activities, including business, learning, entertainment and so on. Forty million user files of MasterCard and VISA were exposed in 2005 when the company cooperating with CardSystem Solutions was hacked [1, 2]. Many people were forced to renew their credit cards to avoid any financial loss. This event shows the importance of network security. Besides, internal hacking is difficult to detect because firewalls and IDSs usually only defend against outside attacks.

Currently, many systems can identify who the user logging into a system is by deploying biotechnical verifications [3-9]. Most current computers check UID and password as an authentication. But hackers may install Trojans to pilfer victims'

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 137–152, 2007.
© Springer-Verlag Berlin Heidelberg 2007

security patterns, or issue a large scale of trials with the assistance of a dictionary to access users' passwords before they can "legally" log in to a system. When successful, hackers may access users' private files or even destroy system settings. Most host-based security systems can discover an intrusion from a user's logged history afterward. And most network-based systems can detect an intrusion online [10-12]. However, to identify who the attacker is in real-time is difficult since attack packets are often issued with forged IPs.

In this paper, we propose a security system, named the Intrusion Detection and Identification System (IDIS), which mines log data to identify commands and their sequences (together named command sequences (C-sequences in short)) that a user habitually submits and follows respectively as the user's forensic features. When an unknown user logs in to a computer, the IDIS starts monitoring the user's input commands to detect whether he/she is issuing an attack. In the following, we use "hacker", "attacker" and "intruder" interchangeably as the same terms are even defined differently by different authors.

The rest of this article is organized as follows. Section 2 introduces the related research. Section 3 describes the framework and details of the IDIS. Experimental results are shown in section 4. Section 5 concludes this paper and addresses our future work.

2 Related Research

Computer Forensics, which views computer systems as scenes of a crime, is computer security technologies that analyze what attackers have done. Most of their applications focus on how to identify malicious network behaviors and the characteristics of attack packets, and the way to identify attack patterns based on their analyses. Abdullah et al. [13] used package dump tools, such as tcpdump and pcap, to collect and analyze network packets and to identify network attacks from different network states and packets' distribution.

Yu et al. [14] provided another example of integrating computer forensics with IDS. A knowledge-based system was deployed to collect forensic features from malicious network behaviors. This system performed excellently in improving the hit rate of intrusion alerts.

Yin et al. [15] proposed an approach that built a Markov chain to describe users' normal operations. A state of the chain records the probability of entering the next state. However, this approach focuses on system calls generated instead of commands submitted. Chau et al. [16] used a pattern extraction technique to identify particular crime data, such as segmenting and extracting a suspect from a picture on a security video. Cabrera et al. [17] deployed sequential pattern mining to identify attack patterns that hackers frequently submit, and classified the *modus operandi* that suspects used in the commission of crimes into predefined crime types.

These techniques and applications truly contribute to network security. However, they cannot easily authenticate remote-login users, and cannot detect specific types of intrusions, e.g., when an unauthorized user logs in to a system with a legal UID and password. Authentication based on the user's operation habits is what we propose.

The IDIS uses data mining and forensic techniques to respectively analyze and identify user operation characteristics, which as a kind of biological characteristics are essential in identifying a user. This system can identify attack patterns that hackers often use as well. By long-term observation, user habits can be effectively identified.

3 System Framework

The IDIS consists of a command monitor, detection server, mining server and honey pot. The command monitor bridges the system kernel and shell, collects input commands from the underlying user and transfers the collection to a detection server, which not only stores these commands in the user's log file, but also compares these commands with attack patterns in real-time to identify the sources of attacks. In the IDIS, attack patterns are represented by a reverse tree, a tree on which commands are organized in the reverse order of their arrival from the root. If the input commands and attack patterns match, the detection server notifies the command monitor to isolate the user from the system kernel and redirects his/her inputs to a honey pot. The honey pot is a virtual system that conceals the protected system from the user and collects many attack patterns to increase the IDIS's detection capability. By deploying a reverse tree, we can lightweight the IDIS and speed up online detection. The mining server analyzes log data with data mining techniques to identify user habits. The IDIS can identify who an underlying user is in a given intranet by comparing the user's current inputs with all others' habits.

3.1 Command Monitor

As an extended portion of an operating system, the command monitor comprises an input interceptor and system call filter. It records sensitive system calls, which are defined as system calls that may change system settings (e.g., kill process and close service) and access sensitive data (e.g., open a file that has system attributes), in a table named a sensitive call table. When a user submits a command, at least one interrupt for receiving and processing the command will be generated. The system kernel on accepting the interrupt or the first interrupt will notify the input interceptor, a subsystem that intercepts user commands, to receive the command and pass it to the shell (command interpreter). A system call generated by the shell in executing a command is compared with a sensitive call table. Once matched, the system call will be transferred to a system call filter to check whether the call is safe or not based on the safety level of the call and the access limitation of the user. Unsafe system calls will be retained for further analysis. Safe ones will be sent to the system kernel to perform their corresponding service.

In fact, many user operations, including mouse movement and key press, will generate interrupts. Our goal is to extract the characteristics of user inputs. Therefore, only commands are addressed in this research so as to simplify the scope.

Besides, we divide users into groups according to their occupations. Each group G has its corresponding inhibited commands, named a class-limited command list (G), e.g., an unauthorized office receptionist should not submit network administrating commands [18].

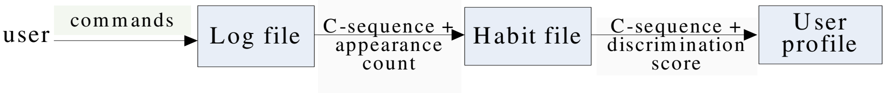

user
commands
Log file
C-sequence + appearance count
Habit file
C-sequence + discrimination score
User profile

Fig. 1. Control flow of the generation of a user profile

3.2 Mining Server

A mining server extracts commands that a user has habitually used from his/her log file, identifies C-sequences, counts the times that a C-sequence appears in the log file, and stores the result in the user's habit file, in which a C-sequence is followed by its appearance count. The user's habit file is then compared with all others' habit files in the underlying system to identify the user- specific C-sequences and C-sequences that are commonly used by all or most users. After that, a C-sequence's discrimination score, used to identify whether the probable user who submits the C-sequence is among the users of the system, is calculated to generate the user's profile [19, 20], which is a copy of his/her habit file, but a C-sequence's appearance count is replaced by its discrimination score. Fig. 1 illustrates the control flow. The way to calculate a discrimination score will be described later.

3.2.1 Mining User Habits and Attack Patterns

A log file consists of many sessions. Each comprises commands a user has submitted within the period of time between his/her login and the corresponding logout. Given a user's log file, the IDIS processes the commands with a size 10 sliding window, named a Log-sliding window (Log-window), to partition the commands along their submitted sequence into k-grams where k is the number of consecutive commands, k =2, 3, 4....10. The reader may ask why the window size is 10. The reason is that according to our data the longest user habit we collected in this research was 6 commands, and the longest command sequence, e.g., L, that contains a user habit, e.g., H, constrained on first(L)=first(H) and last(L)=last(H) was 10 commands where first(X) and last(X) respectively represent the first and last commands of a sequence of commands X where X=L or H. In addition, another sliding window of 10 commands, named a Compared-sliding window (Com-window), is deployed on another concerned session. This time, k' consecutive commands, preserving their submitted sequence, are extracted from a Com-window generating a total of (10 – k' + 1) k'-grams, k'=2, 3, 4,...10. The mining server compares each of $\sum_{K=2}^{10}(10-k+1)$ k-grams with $\sum_{K'=2}^{10}(10-k'+1)$ k'-grams by using the longest common sequence algorithm which can reveal the similarity between two strings by skipping noises. After that, the Com-window shifts one command right. The procedure is repeated until the last session of the log file is involved. Then the Log-window shifts one input command right. The whole procedure is repeated until the last ten or all (if less than ten) commands of the last second session are encountered by the Log-window.

Here is an example. Assume that a log file has r sessions. Each of the first r-1 sessions, e.g., session x, x=1,2,…r-1, is compared with each of its following sessions (i.e., sessions x+1 to r), denoted by session y. That is, a total of $r(r-1)/2$ session comparisons are performed. During the comparison of sessions x and y, each of the |session x|-9 Log-windows will be compared with each of the |session y|-9 Com-windows where |session u| is the number of commands session u has, and u=x or y. Hence, a total of (|session x|-9) • (|session y|-9) window comparisons are performed. On each comparison of two windows, $\sum_{k=2}^{10}(10-k+1)$ k-grams are compared with $\sum_{k'=2}^{10}(10-k'+1)$ k'-grams. Therefore, there is a total of $\sum_{x=1}^{r-1}\sum_{i=1}^{|session\ x|-9}\sum_{k=2}^{10}((10-k+1)\bullet\sum_{y=x+1}^{r}\sum_{j=1}^{|session\ y|-9}\sum_{k'=2}^{10}(10-k'+1))$ k-gram and k'-gram comparisons.

Fig. 2 gives two examples. The solid-line rectangles list two log sessions, sessions r and p. The shaded areas are Com-windows. The dotted-line rectangle contains a C-sequence, a k-gram, extracted from a Log-window on session m where k= 7. "ls", "cat", "ping", "ping", and "telnet" in session r are matched in order to the k-gram at k' = 7. "telnet" and "man" are noises, and thus are ignored. The longest common sequence between the k-gram and k'-gram in sessions m and p, respectively, is "ls", "cat", and "cp".

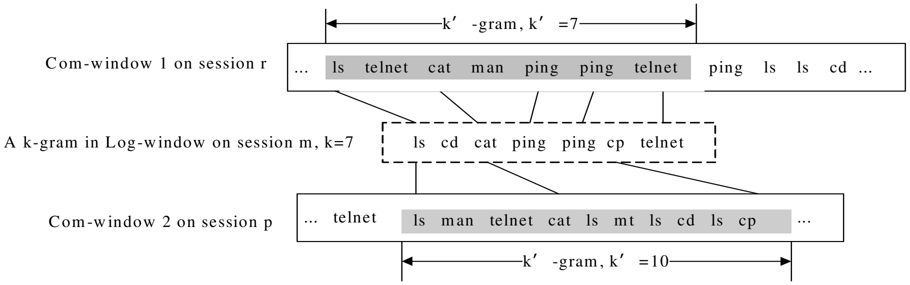

k' -gram, k' =7
Com-window 1 on session r
... ls telnet cat man ping ping telnet ping ls ls cd ...
A k-gram in Log-window on session m, k=7
ls cd cat ping ping cp telnet
Com-window 2 on session p
... telnet ls man telnet cat ls mt ls cd ls cp ...
k' -gram, k' =10

Fig. 2. Two examples of comparisons between a k-gram in a Log-window on session m and a k'-gram in a Com-window (the shaded area) of 10 commands on session p (7 on session r) with the longest common sequence algorithm

Fig. 3 shows an example of a habit file in which a line is a habit, also a C-sequence, ended by its appearance count. The more frequently a C-sequence appears, the higher probability the sequence is the user's habit. After the habit file is constructed, each time the user logs in and logs out, comparison algorithm described above will be invoked under the situation that the current session is treated by a Log-window and log file sessions are processed by a Com-window to generate new habits and to increase habit counts.

Furthermore, we can apply this algorithm to known attackers' log files to extract their usage habits. After legal operations have been ripped off, what remains is attack patterns that form a signature file [12].

```
telnet exit = 98
save telnet exit = 6
save pine = 8
save exit pine = 3
  ...
```

Fig. 3. A part of a user's habit file. Each line is a habit ended by its appearance count.

```
Exit pine exit = 0.559
save exit pine exit = 0.379
save telnet exit pine exit = 0.379
save exit who = 0.550
  ...
```

Fig. 4. A part of a user profile. Each line is a habit ended by a corresponding discrimination score.

3.2.2 Creating User Profiles

As stated above, a user's profile is a habit file, but each habit is ended by a discrimination score instead of an appearance count. Let DSc_{ij} be the discrimination score of C-sequence j, a usage habit, submitted by a user i.

$$DSc_{ij} = \frac{H_{ij}}{\sum_{t=0}^{n-1} H_{tj}} \tag{1}$$

where n is the number of users in the intranet concerned, and H_{ij} is C-sequence j's appearance count in user i's habit file. DSc_{ij} is a floating number ranging from 0 to 1 for all i and j. A user's habitual C-sequence, that rarely appears or has not appeared in others' habit files, will obtain a high score. Those given low scores are commonly used C-sequences. Fig. 4 shows a part of a user profile.

An attack pattern which is hacker-specific or commonly used can be identified in the same way. However, only hackers' habit files (or a signature file) and illegal C-sequences are considered. Similarly, an attack pattern that a hacker frequently submits but others have seldom or never submitted should be one of the hacker's representative attack patterns, and is thus given a representative high score. Those obtaining low scores are attackers' common behaviors which can help to determine quickly whether a set of input commands is an attack or not by comparing these commands with attackers' common behaviors. Hacker-specific patterns can help to identify who the hacker is.

3.2.3 Similarity Scores

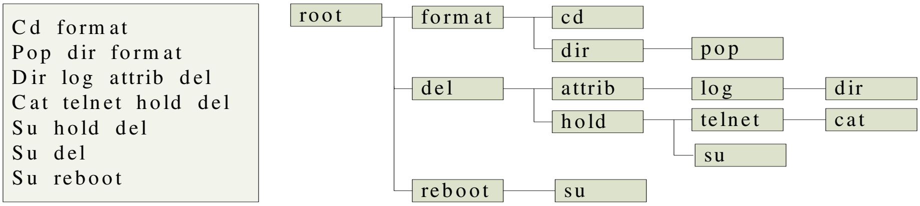
We deploy formula (2) [21], which is a formula commonly used to assign a weight to a term in the information retrieval domain, e.g., to calculate a weight for a C-sequence. Given a set of user habit files $D = \{UP_1, UP_2, ...UP_N\}$ where N is the number of users (also, number of habit files) in an intranet, let $T = \{CS_1, CS_2, ...CS_k\}$ be the set of C-sequences retrieved from D, and $D_i = \{UP'_1, UP'_2, ...UP'_{Mi}\}$ the set of habit files that contain CS_i, $CS_i \in T$. The weight W_{ij} of CS_i in UP_j is defined as

$$W_{ij} = \frac{\mathrm{sf}_{ij}}{\mathrm{sf}_{ij} + 0.5 + 1.5\,\frac{ns_j}{AVG(ns)}} \times \frac{\log\left(\frac{N+0.5}{M_i}\right)}{\log(\mathrm{N}+1)} \quad \mathrm{i}=1,2,3...\mathrm{M_i},\ \mathrm{j}=1,2,3...\mathrm{N} \quad [21] \tag{2}$$

where sf_{ij} is the appearance count of CS_i in UP_j, ns_j the total number of C-sequences in UP_j, $AVG(ns)$ the average number of C-sequences an element of D has, and log((N+0.5)/M$_i$)/log(N+1) is the ICPF (inverse characteristics profile frequency) [21]. Given an unknown user x's current input commands CMDs (1≦x≦N), the similarity score between CMDs and UP$_j$ is defined as $SimS_{xj} = \sum_{i=1}^{p} W_{ij}$ where p is the number of C-sequences appearing in both the CMDs and UP$_j$. The higher a similarity score, the higher probability the user is the person who submits these input commands.

3.3 Detection Server

As stated above, attackers' common behaviors are represented by a reverse tree, named common reverse tree. Fig. 5 gives an example. Nodes of the first level, including format, del and reboot, are the last commands of attackers' common behaviors. All hacker-specific attack patterns are also organized in a reverse tree, denoted by specific reverse tree, in which a leaf node is accompanied by the corresponding hacker's ID. A monitored command list consists of two lists, a common monitored command list and a specific monitored command list, which respectively comprise all commands of the first levels of the common and specific reverse trees.



(A) Attack patterns **(B)** The corresponding common reverse tree

Fig. 5. Representation of attackers' common behaviors

Assume that a user has entered m commands {C$_1$, C$_2$,…C$_m$} after his/her login where C$_i$ is submitted prior to C$_{i+1}$, i = 1, 2, … , m-1. To detect intrusion online, the underlying input command C$_i$, 1≦i≦m, is compared with the common monitored command list first. If they match, a modified deep-first search traversal algorithm will start traversing the sub-tree of the common reverse tree which is rooted at C$_i$ to determine whether the user's inputs, C$_j$ to C$_{i-1}$, j = 1, 2,….or i-2, in the reverse order of their arrival can finally reach a leaf node or not.

In this algorithm, when one input command C$_j$ matches node N$_k$, a node at level k, in the common reverse tree, and C$_{j-q}$, q = 1, 2, …, or j-1, with the smallest q (if there are several such C$_{j-q}$) also matches one of N$_k$'s immediate child nodes, e.g., N$_{k+1}$, then let C$_j$ = C$_{j-q}$ (i.e., skip q-1 consecutive noisy input commands) and try to find another C$_{j-q}$ that matches one of N$_{k+1}$'s immediate child nodes. Each time when no such C$_{j-q}$ can lead the algorithm to arrive at a leaf, the algorithm backtracks to the last second matched node, e.g., N$_k$ (note the last matched one is N$_{k+1}$), and tries to find C$_{j-q'}$ that matches N$_k$'s other immediate child where C$_{j-q'}$ is located between C$_1$ and C$_{j-q}$ (since on the last match on N$_k$, the smallest q is chosen as the next command that matches

N_{k+1}, therefore, it should be that q'≧q), and q' is the smallest integer if several q's exist. When one is found (at level k+1), the algorithm looks for a matched node at level k+2. The procedure is repeated until no such C_{j-q} can match N_k's immediate child or the algorithm reaches a leaf node.

The latter is considered as an attack. The detection server replies to the command monitor with an "unsafe" message. The former will trigger the IDIS to perform the algorithm again but, this time, on the specific reverse tree instead of on the common reverse tree to check whether there is a hacker-specific attack or not. If yes, the detection server replies with an "unsafe" message accompanied by the hacker's ID; otherwise, it sends a "safe" message to the command monitor.

3.4 Attack Types

In this study, there are three types of intrusions. Type 0 is defined as the situation where a user of a specific group submits a command that the group members are prohibited from using. Type I attack is an attack that intrudes into a system using a sensitive command, which is defined as a command that erases or modifies sensitive data or system settings, to change the environmental settings of the system, or to attack the system. A type I command is often preceded by several legal and safe commands. A type II attack consists of several attack patterns, and each can be considered as a stage. In fact, a hacker mixing specific commands with commands of an attack pattern can sometimes successfully penetrate a security system.

Generally, a command that generates a buffer-overflow attack is a normal command with a parameter longer than legal length. With this kind of attack, a hacker can shut down the target system with one command, or grant higher access right to further his/her intrusion, e.g., download passwords. The former belongs to type I. The latter is type II since it has at least two stages, e.g., grants a higher access right and then attacks the system. Hence, a type I command is also called a single command attack pattern. The commands to launch a type II intrusion are called a multi-stage attack pattern. Basically, a single command attack pattern is a special multi-stage attack pattern when the number of stages = the number of commands = 1.

Type 0 attacks can be detected by comparing an input command with a class-limited command list. However, this detection cannot protect a system completely, especially when a user is able to switch from one account to that of a user in another group.

3.5 Honey Pot

To prevent hackers from realizing that they are facing a virtual environment and to protect the concerned environment, a honey pot should provide a platform similar to the protected one, and safeguard its own disk and kernel since all programs and data that a hacker wants to access are saved and handled by the two subsystems, respectively. Also, a honey pot collects attack patterns in its own log file.

3.5.1 Disk Subsystem

Initially, the IDIS duplicates the contents of a protected server's hard disks to a honey pot. The operating systems of the server and honey pot are the same. Once the server is under attack, the IDIS switches control to the honey pot in which disk files are all in read-only mode, and all operations onto disks are redirected to a ram disk. To achieve this, we re-write disk and file service routines, and replace the original service routine of a system call in a system call table with a corresponding routine of our own, which performs I/O and file operations on ram instead of on disk and file.

The first time a file is accessed by an attacker, the honey pot's file service routine retrieves the corresponding file description information from the physical disk subsystem, and records it in a virtual file table, a table constructed in ram disk to map file paths to their contents. Table 1 shows an example in which /usr/dz0/setting.txt after being loaded to ram disk is placed in frames 1, 2 and 4. The table contains a deletion flag for each path to indicate whether the path has been deleted or not. All file contents that have been modified are also saved in ram disk without having their physical disk contents updated.

When an attacker deletes a file, only memory is released. The corresponding record in the virtual file table remains, with deletion flag= true. The second record/root/command.log in Table 1 is an example. Frames 3, 5, 6, and 7 were given back to the operating system. If a hacker tries to access the file again, a file service routine based on the record will not reload the file to the ram disk and the hacker will assume that the file has been deleted.

Table 1. An example of a virtual file table

File path	Frame # in ram disk	Deletion flag
/usr/dz0/setting.txt	1, 2, 4	false
/root/command.log	3, 5, 6, 7	true
…	…	…

Valuable files, e.g., account books and documents regarding special techniques, and important system settings, e.g., boot, network and firewall setting files, are often hackers' targets. Hackers access the information for specific purposes (e.g., stealing business secrets) or modify these settings to set up a back door for future intrusion. Hence, for each important file, an understudy is created as a hidden file, and is placed on the same path as that of the important file with the same file name but extended by "uds", e.g., an important file, named config.sys, of which the understudy is config.sys.uds. Further, file contents of an understudy are produced by randomly generating or providing fake contents to prevent the original contents from being stolen.

On a hacker's first access to an important file, the command monitor looks up the corresponding record in the virtual file table based on the given path, and checks to see whether an understudy exists or not. If yes, the command monitor loads the file instead of the original from the disk. Fig. 6 shows the control flow.

The reader may assume that if a false positive occurs, what the authorized user has done will be ignored, and the updated data will remain unchanged. We solve this problem by saving all the updated files and data in a honey pot with a file extension, e.g., .susp, and redo all operations if necessary, e.g., after the updated files and data have been authenticated by the system manager. If no one complains that his/her operations have been ignored, the saved files and data will be deleted after a pre-defined period of time.

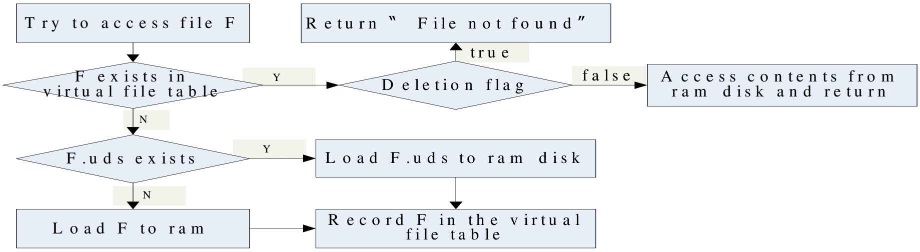


Fig. 6. File accessing flow in a honey pot

3.5.2 System Kernel

In a single-user multi-tasking system, e.g., MS-Windows operating system, only the disk system needs to be protected. All modifications to ram disk performed by hackers will disappear after system shutdown. The system kernel and settings remain in the same state as at the time point when the command monitor switched the hacker's inputs to the honey pot. Rebooting can recover them.

In a multi-user multi-tasking system, both disk and kernel need to be protected. There are two ways to implement a honey pot as a multi-user multi-tasking system. In the first approach, an extra machine such as a mirror is required. All user operations, including submitting and executing commands, key strikes and mouse moving and clicking, are sent to both the mirror and the protected system, but responses generated by the mirror are suppressed. When an attack is discovered, the hacker's inputs are transferred to the honey pot (i.e., the mirror) only, not to the protected system. System responses now come from the honey pot. The advantage is that normally the honey pot is an online system, and the time required to switch an attacker from the protected system to the mirror is almost zero. Attackers will unsuspectingly face the honey pot. The disadvantage is that an extra machine is required.

In the second approach, the honey pot is implemented as a virtual machine platform, which tracks user operations normally. When the command monitor discovers that there is an attack, it notifies the honey pot to build a virtual machine for each hacker, transfers memory image generated for the hacker to the virtual machine, and then redirects the hacker's inputs to the virtual machine. The disadvantage is that it requires transferring memory image to the honey pot, causing extra network flow, e.g., for a 32-bits system, probably up to 4 GB data have to be transferred. An attacker may issue several attacks at the same time to generate a mass of inner network flow (i.e., a DoS/DDoS attack) which will paralyze the system, thus preventing it from performing normal services.

Fig. 7 shows our framework, mirror twin, which is a compromise between the two approaches, and in which the mirror is a virtual machine platform. Both the honey pot and the backup system of the protected system are virtual systems running on the platform. The mirror mirrors the protected system like the first approach stated above.

When an attack is discovered, the command monitor running on the mirror creates a virtual machine for the hacker and moves his/her memory image to the virtual machine. All the hacker's inputs are redirected to the virtual machine to continue serving the hacker. The hacker now enters a honey-pot mode from a protection mode. Meanwhile, the command monitor in the protected system kills the process originally generated for serving the hacker and notifies the router (firewall) to stop routing the hacker's input commands to the protected system. Fortunately, due to limitation of memory by the operating system (i.e., virtual memory), the size of a memory image is seldom huge. However, once too much network flow is issued between the protected system and the mirror, the firewall will disconnect the sessions established for the hackers to avoid an internal DoS/DDoS attack.

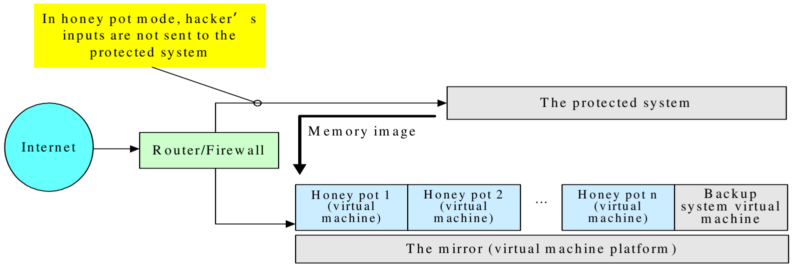

In honey pot mode, hacker' s inputs are not sent to the protected system
Internet
Router/Firewall
The protected system
Memory image
Honey pot 1 (virtual machine)
Honey pot 2 (virtual machine)
...
Honey pot n (virtual machine)
Backup system virtual machine
The mirror (virtual machine platform)

Fig. 7. Framework of a mirror twin

4 Experiments

We collected 655 students' log files between October 1, 2006 and January 31, 2007 from the Tunghai University Computer Center as the experimental data. None of the students was told that his/her computer habits would be observed. Hence, they all acted normally. Among the students, 424 were computer science students, and 231 were in the Management and Social Science colleges. Three characteristics of user habits were examined, including commands that were frequently used, the order in which they were submitted, and commands that users mistyped. Interestingly, some logged arguments are much more user representative than commands. According to our observation, mistyped commands submitted by sophisticated users are powerful tools for distinguishing one user from another, e.g., several students very often type "cpoy", "dell" and "form" instead of typing "copy", "del" and "from" (an SQL statement), respectively. Hence, we decided to keep them in history files for future research.

All commands extracted from a user log file were saved in the user's habit file in accordance with their originally submitted sequence. Other parameters, e.g., time and date, were removed to simplify the scope of the following experiments. To perform privacy preserving experiments [22], we hid user IDs for all users in the data set involved.

4.1 Identifying Unknown Users

The first experiment, identifying an unknown user, was performed ten times, and each time we selected a different 75% of commands from each log file as the training data to generate his/her profile. The remaining 25% were the test data (a test file). Given an unknown user's test data, if a user's similarity score, the average of its ten-fold

Table 2. A portion of experimental results generated by a cross comparison by comparing each user's test data (25% of a profile) with 655 user profiles (75%). (sXX28XX and sXX29XX are computer science students).

	User's test file	Rank	Decisive rate %	Cost (msec)
1	sXX2808.tst	0	100.00	34
2	sXX2959.tst	77	90.12	382
3	sXX2811.tst	165	79.43	331
…	…	…	…	…
108	sXX2849.tst	7	102.15	67
…	…	…	…	…
655	sXX3103.tst	15	104.73	839
A total of 655 user profiles (75%) Avg._rank = 5.51 Total cost = 53719 msec			A total of 655 test files (25%) Average decisive rate = 99.16%	
Average number of commands per user profile = 166.24 Average number of commands per test file = 41.56				

values, is within the first x% of all users, we say the decisive rate is x (=(|UPs|-avg_rank)/|UPs|*100%)%, $0 \le x \le 100$, where |UPs| is the number of user profiles, and avg_rank, the average of all users' ranks, is defined as $\frac{\sum_{i=1}^{|UPs|}(\sum_{j=1}^{q} rank_{ij} / q)}{|UPs|}$ where q (=10) is the number of the experiment and $rank_{ij}$ is the average of user i's ten-fold ranks. The experimental result, shown in Table 2, indicates that avg_rank=5.51, and the average decisive rate x of all users was 99.16% (= (655-5.51)/655*100%). "Rank P" and "Cost Y ms" respectively show that a user profile is ranked P and the time required to compare the user's test file (i.e., 25%) with 655 profiles (i.e., 75%) is Y ms.

Students not majoring in computer science often use simple, common commands. Many of their logged commands are submitted under the guidance of sophisticated users, e.g., teachers or highly computer literate students. Their log sessions are often short and highly similar. Thus, the average decisive rate of these students was 94.43%.

Also, we implemented the algorithm proposed by [15] to run the experimental data again. Avg_rank= 32.54, and the decisive rate was 95.03%. For non-computer science students, the decisive rate was 90.82.

4.2 Detecting a Single Command Attack Pattern

To detect a single command attack pattern, we randomly inserted twenty different attack commands AC1 to AC20 into a legal user's log file. Before retrieving the user's habits, we ripped off three types of patterns including those that are legal and safe, that contain more than one attack command, and that are not ended by an attack command. We deleted the third because those located after an attack command are safe according to the definition of a single command attack pattern. In addition, if there is a pattern S = "ls cat AC1 vi ls" in a user profile, then in this profile there must exist two patterns “ls cat AC1” and “ls cat AC1 vi” which are subsets of S, forming redundant patterns.

After that, we selected a% of user log contents from each log file and b% of collected attack patterns, and mixed them to simulate attackers’ inputs, which will be described below.

From 655 user log files, we extracted 108,888 commands, from which 2012 attack patterns were identified. Let a = b = 15 and a size-ten sliding window is used to mix an attack pattern and legal commands. Table 3 shows the mixed patterns and the detection results. The field “ID of an inserted pattern (inserted location)” lists IDs of single command attack patterns followed by their locations. They were generated by the following procedure. Given an attack pattern ($C_{1,} C_2, \ldots , C_x$) where C_i is ith command of the pattern, the last command C_x was first inserted into the corresponding session of a given log file at position q as the qth command. The remaining x-1 commands were mixed with the leading 10-x log file commands R={ $C'_{q-(10-x)}$, $C'_{q-(10-x)+1}$,…, C'_{q-1}} in the underlying session by randomly inserting C_i, i = 1, 2, 3, …x-1, into any position among the elements of R under the constraint that C_i should be prior to C_{i+1}. After insertion, C_i might be adjacent to C_{i+1} or separated by several commands, and now the location of C_x became q+x-1. For example, #793(11) in the first record of Table 3 indicates that the last command of attack pattern #793 (cd cd ls AC2), x=|#793| = 4, and is inserted into the 8th (i.e., qth) position of log 01982. Original C'_8 becomes C'_9. Commands cd, cd and ls are mixed with those in {$C'_{2,} C'_3$, … , C'_7}. After that, C_x becomes the 11th command. Nevertheless, the commands coming from both #793 and R individually follow their original submitted sequence.

Table 3. Patterns obtained by mixing log contents with a single command attack pattern and the detection results

	Mixed patterns		Detected attack patterns	
User log file	No. of inserted patterns	ID of an inserted pattern (inserted location)	No. of detected patterns	ID of a detected pattern (detected location)
log01982	2	#632(4), #793(11)	2	#2000(4), #1657(11)
log02864	19	#7(5), #1603(12), …	19	#1(5), #100(12), #2001(20),
log02871	9	#1515(6), #1845(15), …	9	#1991(6), #2006(15), …
…	…	…	…	…

The field “ID of a detected pattern (detected location)” in Table 3 represents the ID of a detected pattern followed by a detected location, which is the position of the pattern’s last command. For example, #1657(11) in the first record shows that attack pattern #1657 (ls AC2), which ended at position 11 of log01982, was detected. Basically, a shorter pattern may be contained in a longer one. Hence, the ID of a detected pattern may be different from that of the one inserted since #1657 is a proper subset of #793. From Table 3, we can conclude that the number of patterns inserted is equal to that of those discovered, i.e., precision = recall = 100%. The detection process costs 326 ms, which is short enough to detect attacks in real-time, i.e., the two reverse trees can effectively help users to identify hacking behaviors.

```
A1_Stage_1 A1_Stage_2 A1_Stage_3
A2_Stage_1 A2_Stage_2
A3_Stage_1 A3_Stage_2 A3_Stage_3 A3_Stage_4
…
```

Fig. 8. A part of a log file consisting of multi-stage attack patterns. Each line has several stages. Each stage is a user habit ended by a single command attack pattern.

Table 4. Patterns obtained by mixing log contents and multi-stage attack patterns, and the detection results

	Mixed patterns		Detected attack patterns	
User log file	No. of inserted patterns	ID of an inserted pattern (inserted location)	No. of detected patterns	ID of a detected pattern (detected location)
log12815	7	#1(4), #3(13), #4(20)…	7	#1(4), #3(13) , #4(20)….
log12826	25	#1(7), #6(16), #8(26)…	25	#1(7), #6(16), #8(26),…
log12838	7	#6(6), #9(15), #2(22)…	7	#6(6), #9(15), #2(22)…
...	...	...	...	...

4.3 Detecting a Multi-stage Attack Pattern

Fig. 8 shows a log file consisting of multi-stage attack patterns. Each line has several stages. Each stage in turn is a user habit ended by a single command attack pattern. For detecting multi-stage attack patterns, again, let a = b = 15 given nineteen stages, S1 to S19, from which 85 different attack commands are retrieved, i.e., $|\bigcup_{i=1}^{19}\{C_r \mid C_r \textit{ is a command}, C_r \in S_i\}| = 85$ where $\{C_p \mid C_p \textit{ is a command}, C_p \in S_i\} \cap \{C_q \mid C_q \textit{ is a command}, C_q \in S_j\} = \phi$, i, j = 1, 2, …19, and i $\neq$ j. That means, S1 to S19 are non-cross-reference patterns so that all the IDs of detected patterns, as shown in Table 4, are exactly the same as those of the inserted ones. The two reverse trees are both six levels in height, and 108,888 commands extracted from 655 user log files were compared, costing 469 ms which is also short enough to discover attacks in real time and which is also longer than that for detecting single command attack patterns since reverse trees are higher and tree traversal time, of course, is longer. Also, the precision = recall = 1.

5 Conclusions and Future Work

In this article, we proposed an approach to find out users' habits deploying data mining and forensic techniques. To identify the representative C-sequences for a user, the times that a habitual C-sequence appears in the user's log file is counted and its discrimination score is calculated so that the user's profile can be established. By comparing a user's current input commands with all others' profiles, the IDIS can identify who the user is. The accuracy is high enough to make the IDIS a valuable auxiliary subsystem in a closed environment where it can assist in the identification of an internal hacker. Of course, a new user whose user profile has not been established will not be a candidate for identification. Meanwhile, a user's input commands are compared with two reverse trees in which all commands of an attack pattern are organized in their reverse order so as to real-time detect whether underlying inputs are attacks or not. By comparing these inputs with specific reverse trees, we can further identify who the hacker is. Attack patterns are extracted from attackers' log files by ripping off legal operations. Employing reverse trees can lightweight IDIS and lower the load of the detection server.

Moreover, accurately and completely collecting user behaviors on much more basic operations, such as system calls instead of commands, is much more helpful in detecting hackers and identifying a user. Such procedures will also help to collect intrusion behaviors in a system that employs GUI interface.

However, how to process and mine such a huge volume of data may be the first challenge. Several papers have addressed this topic [15, 23]. But many systems have not been implemented, and many did not describe their implementation. Additionally, to detect an attack and respond in real time, we need a fast algorithm and a distributed computing environment to speed up data processing since the time complexity of the algorithm that compares k-grams and k'-grams is high--about $O(n^6)$. Cluster and/or Grid computing should be the best candidates. A mathematical analysis of the IDIE's behaviors which would lead to the building up of its formal performance and cost models may be interesting. As stated above, the records of mistyping are also helpful in identifying underlying users, and can increase detection accuracy and improve the decisive rate. These are our future research topics.

References

[1] Schneier, B.: CardSystems Exposes 40 Million Identities. http://www.schneier.com/blog/archives/2005/06/cardsystems_exp.html

[2] Mitchell, A.P.: 40 million. http://www.theinternetpatrol.com/cardsystems-compromises-data-of-40-million-mastercard-and-visa-cardholders

[3] Sheng, Y., Phoha, V.V., Rovnyak, S.M.: A Parallel Decision Tree-Based Method for User Authentication Based on Keystroke Patterns. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 35(4), 826–833 (2005)

[4] Ray, M., Meenen, P., Adhami, R.: A Novel Approach to Fingerprint Pore Extraction. In: IEEE Southeastern Symposium on System Theory, pp. 205–208 (March 2005)

[5] Krašnjak, D., Krivec, V.: Fingerprint Classification Using a Homogeneity Structure of Fingerprint's Orientation Field and Neural Net. In: International Symposium on Image and Signal Processing and Analysis (2005)

[6] Yuan, W., Lin, Z., Xu, L.: A Rapid Iris Location Method Based on the Structure of Human Eyes. Engineering in Medicine and Biology Annual Conference (September 2005)

[7] Meenen, P., Adhami, R.: Approaches to Image Binarization in Current Automated Fingerprint Identification Systems. In: IEEE Southeastern Symposium on System Theory (March 2005)

[8] Nanni, L., Lumini, A.: A Deformation-invariant Image-based Fingerprint Verification System. NeuroComputing 69(16), 2336–2339 (2006)

[9] Antonelli, A., Cappelli, R., Maio, D., Maltoni, D.: Fake Finger Detection by Skin Distortion Analysis. IEEE Transactions on Information Forensics and Security 1(3), 360–373 (2006)

[10] http://www.snort.org

[11] http://en.wikipedia.org/wiki/Snort_%28software%29

[12] Leu, F.Y., Lin, J.C., Li, M.C., Yang, C.T.: A Performance-Based Grid Intrusion Detection System. In: International Computer Software and Applications Conference (2005)

[13] Abdullah, K., Lee, C., Conti, G., Copeland, J.A.: Visualizing Network Data for Intrusion Detection. In: the IEEE Workshop on Information Assurance Workshop (June 2005)

[14] Yu, J.Q., Reddy, Y.V.R., Selliah, S., Kankanahalli, S., Reddy, S., Bharadwaj, V.: TRINETR: An Intrusion Detection Alert Management System. SIPLab, Concurrent Engineering Research Center Lane Department of Computer Science and Electrical Engineering, pp. 235–240 (June 2004)

[15] Yin, Q., Shen, L., Zhang, R., Li, X.: A New Intrusion Detection Method Based on a Behavioral Model. In: World Congress on Intelligent Control and Automation, pp. 4370–4374 (June 2004)

[16] Chau, M., Xu, J.J., Chen, H.: Extracting Meaningful Entities from Police Narrative Reports. In: National Conference on Digital Government Research, pp. 271–275 (2002)

[17] Cabrera, J.B.D., Lewis, L., Mehra, R.K.: Detection and Classification of Intrusion and Faults Using Sentences of System Calls. SIGMOD Record 30(4), 25–34 (2001)

[18] Dridi, F., Muschall, B., Pernul, G.: Administration of an RBAC System. In: International Conference on System Sciences, pp. 1–6 (2004) http://csdl2.computer.org/comp/proceedings/hicss/2004/2056/07/205670187b.pdf

[19] Okazaki, Y., Sato, I., Goto, S.: A New Intrusion Detection Method Based on Process Profiling. In: Symposium on Applications and the Internet, pp. 82–90 (February 2002)

[20] Dickerson, J.E., Dickerson, J.A.: Fuzzy Network Profiling for Intrusion Detection. In: International Conference of the North American Society on Fuzzy Information Processing, pp. 301–306 (July 2000)

[21] Leuski, A.: Evaluating Document Clustering of Interactive Information Retrieval. In: ACM CIKM, pp. 33–40 (November 2001)

[22] Verykios, V.S., et al.: State-of-the-Art in Privacy Preserving Data Mining. SIGMOD Record 33(1), 50–57 (2004)

[23] Lu, K., Chen, Z., Jin, Z.: An Adaptive Real-Time Intrusion Detection System Using Sequence of System Call. IEEE Electrical and Computer Engineering, 789–792 (2003)

Run-Time Randomization to Mitigate Tampering*

Bertrand Anckaert[1], Mariusz Jakubowski[2], Ramarathnam Venkatesan[2], and Koen De Bosschere[1]

[1] Ghent University, Electronics and Information Systems Department
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
{banckaer, kdb}@elis.UGent.be

[2] Microsoft Research, Cryptography and Anti-Piracy Group
One Microsoft Way, Redmond, WA 98052, USA
{mariuszj, venkie}@microsoft.com

Abstract. The problem of defending software against tampering by a malicious host is not expected to be solved soon. Rather than trying to defend against the first attack, randomization tries to minimize the impact of a successful attack. Unfortunately, widespread adoption of this technique is hampered by its incompatibility with the current software distribution model, which requires identical physical copies. The ideas presented in this paper are a compromise between distributing identical copies and unique executions by diversifying at run time, based upon additional chaff input and variable program state. This makes it harder to zoom in on a point of interest and may fool an attacker into believing that he has succeeded, while the attack will work only for a short period of time, a small number of computers, or a subset of the input space.

1 Introduction

Protecting software against attacks from the outside is a problem that has been largely solved in theory. In practice, however, vulnerabilities continue to be discovered at an astonishing rate. Buffer overflows, for example, were a solved problem as early as the 1960s, yet continue to be the most common type of security issue [19].

Due to the complexity of modern software and the increasing body of legacy code, this and other types of vulnerabilities continue to exist. Run-time randomization acknowledges this and tries to mitigate attacks at a different level: by removing predictability and consistency between different executions. Address space layout randomization (ASLR), for example, is an acknowledgment that buffer overflows and related types of attack will continue to emerge. ASLR is available for mainstream operating systems such as Linux (PaX) and Windows Vista.

* This work is partially funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT).

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 153–168, 2007.
© Springer-Verlag Berlin Heidelberg 2007

Protecting software against a malicious host is sometimes a theoretically unsolvable problem [3]. Intuitively, any protection scheme other than a physical one depends on the operation of a finite-state machine. Ultimately, given physical access, any finite-state machine can be examined and modified at will, given enough time and effort [8]. This intuition is confirmed by many examples: Users cheat in games, DRM systems are compromised, software is installed and used without the proper license, pay-TV suffers from piracy, etc.

Most defenses against malicious hosts are about delaying the first attack. The success of these techniques varies in terms of the additional time and effort required by a tamperer. However, no actively attacked protection has remained unbroken for an extended period of time. Randomization is a promising additional layer of defense: Rather than trying to postpone the first attack, it is about limiting the impact of a successful attack in space and time. However, surprisingly little research is publicly available on randomization against malicious-host attacks.

Existing proposals for randomization against malicious hosts randomize the program before distribution. Unfortunately, diverse copies are in conflict with the current software-distribution model, which requires identical copies to leverage the near-zero marginal cost of duplication. Not surprisingly, commercial implementations of this technique can be found in situations where a network connection can be assumed to distribute the copies digitally: DRM implementations for on-line music stores and digital broadcasters [27]. We suggest combining the best of both worlds by introducing diversity after distribution.

The ideas are discussed against a specific model of a tamperer's behavior: the locate-alter-test cycle (Section 2). It has been long understood implicitly that there are many similarities between tampering and debugging. In this model, we make these similarities explicit. As a result, the techniques presented to counter tampering leverage known difficulties from the domain of debugging: non-deterministic behavior (from the viewpoint of the program) and the fundamental limitations of testing (for every input and every environment). Despite originating from a specific model, the techniques increase the workload to create a fully functional patched version in a more general attack model, which assumes only that behavioral changes are made by modifying the program itself.

In this line of work, we make the run-time execution of the code unique, based upon additional chaff inputs (such as time, hardware identifiers, etc.) and variable program state, including additional fake input dependencies. The goal is twofold: (i) To make it harder for an attacker to zoom in on a point of failure and (ii) to limit the impact of a successful attack to a short period of time, a particular computer, a subset of the input space, etc. The underlying ideas are that (i) an attacker typically repeats the execution of the program with a particular input and slowly zooms in on the part where he thinks a vulnerability may occur. This becomes harder if the execution cannot be replayed at will, and (ii) if we can fool an attacker into believing that he has succeeded for a longer period of time, we can delay the feedback-loop of software tampering. These goals and high-level ideas are motivated in Section 3.

On a lower level, the technique requires a number of basic operations. Some of these operations have been dealt with extensively in academic literature. This paper contains a discussion of operations which have received less attention: (i) a concealed way to augment the user-observed input with chaff input, (ii) a criterion to select fake state and input dependencies and (iii) a diversifier to generate syntactically different, yet semantically equivalent pieces of code. These operations are discussed in Section 4-5. An experimental evaluation of the diversity that can be achieved by a practical implementation and the cost of these techniques in terms of code size and execution time is given in Section 6. Related work is the topic of Section 7, and conclusions are drawn in Section 8.

2 Low-Level Debugging Versus Tampering

Debugging and tampering are similar in many respects: many of the same techniques and tools are used in both disciplines. Debugging software is about finding and reducing the number of defects in a computer program to make it behave as the software provider intends. Likewise, tampering is about finding and reducing the number of undesired features to make it behave as the user desires.

The incentive to tamper with software thus originates from the difference between the behavior intended by the software provider and the behavior desired by the user. This difference can take on many forms; e.g.:

- Some software does not want to install without a valid license key. To some users, this is undesired behavior.
- Software may prohibit the printing of certain documents if a user does not have the right privileges. Many users find this cumbersome.
- Gamers may find it annoying that, e.g., they cannot see through walls, or that their health decreases when they get shot.
- Many users do not want their evaluation version to stop working after the evaluation period.
- Some people find it annoying that their credit card gets charged when they listen to music in a digital container, or when they watch pay TV.

Put another way, debugging is about transforming the semantics encoded in the program to the semantics intended by the software provider. Tampering is about transforming the semantics encoded in the program to the semantics desired by the user. Therefore, it should be no surprise that both disciplines are alike. Many tools, such as IDAPro and SoftICE, and many techniques, such as breakpoints and slicing, have been originally designed for debugging, but are heavily used in tampering. The main difference is that during debugging, a higher-level representation of the program is often available (source code, specification, etc.), while tampering typically starts from machine code or bytecode.

Similar to the edit-compile-test cycle of debugging, tampering is typically a cyclic process. Since tampering is usually done at a low level, the compile phase can be eliminated. Furthermore, we can split up the edit phase, leading to the following cycle:

1. **Locate the origin:** To turn the observed undesired behavior into desired behavior, a tamperer first needs to find the origin of the undesired behavior. For example, the displayed health of a gamer is only a manifestation of the internal state. Locally changing the code that displays his health will not result in the desired behavior: He needs to trace it back to where the internal representation of his health actually gets decreased.
2. **Alter the behavior:** Once the origin is determined, a tamperer needs to determine and apply a set of changes that will alter the undesired behavior into desired behavior.
3. **Test:** In this phase, the tamperer checks if the behavior of the software is as desired. If so, his work is done. Otherwise, more cycles are required.

3 Slowing Down the Locate-Alter-Test Cycle

If tampering is similar to debugging, we can argue along the same lines that making tampering harder is the opposite of making debugging easier.

One of the key concepts in making software easier to debug and maintain is modular design. Such design facilitates local changes and thus minimizes the need to verify the impact of a local change on other parts of the program. Most tamper-resistance techniques [6,7,17] have focused on doing the opposite: making the program more inter-dependent. Existing techniques are thus about **slowing down the alter phase** by requiring an understanding of a larger portion of the program and more binary changes to possibly unrelated sections of the program to effect a small change in the behavior of the program.

In this paper, we focus on slowing down the locate and test phases.

Slowing down the Locate Phase. Looking again at debugging, the first task when dealing with a bug report is to reproduce the problem. This is vital, since one cannot observe a problem and learn new facts if one cannot reproduce it. Furthermore, it is essential to find out if the problem is actually fixed. Reproduction is one of the toughest problems in debugging. One must recreate the environment and the steps that led to the problem [26].

Similarly, reproducing undesired behavior is indispensable for tampering. The manifestation of undesired behavior needs to be traced back to its origin. Typically, a tamperer repeatedly executes the application with a particular input and slowly zooms in on a part where he thinks the undesired behavior may originate.

This requires that execution can be replayed at will. We try to hamper this process by choosing between different control paths based on pseudo-random numbers, timing results, thread scheduling, etc.

In software tamper-resistance, the “bugs” are features that we want to manifest every time, so it seems illogical to make their appearance non-deterministic. We can, however, make sure that these features manifest themselves in different ways by duplicating parts of the program, diversifying them and choosing more or less randomly among the alternatives at run time. This makes it harder for a tamperer to zoom in on the vulnerable part of the program, since the semantics of the program may be constant, but the execution paths will not be identical.

Slowing down the Test Phase. Testing is also a major issue in debugging and software maintenance. It is very hard to foresee every input, every environment, every usage scenario and every combination of applications [12]. Testing can show only the presence of undesired behavior, not its absence.

The techniques discussed in this section increase the number of tests required to manifest all occurrences of the undesired behavior. The underlying idea is that the impact of a successful patch for a small subset of the input space, for a limited number of computers or for a short period of time does not pose a great threat to the software or content provider.

The time required to create a fully functional tampered version of the software is increased by letting the tamperer believe that he has succeeded, while it works only for a subset of the input space, or for a short period of time. Tamperers often work by trial and error. Using incomplete knowledge about the program, they change parts, hoping that the desired results will arise. When it is easy to evaluate whether these results have been obtained, this process can be repeated many times. If this evaluation takes longer (e.g., because it works for most of the input sets most of the time), the workload increases.

Furthermore, the credibility of the tamperer in the cracker community may decrease if he claims to have successfully patched a program, while it still behaves as intended by the software provider on other computers.

We could for example use one type of license check in 90% of the cases and another one in the remainder. This way, the tamperer may be fooled into believing that he has succeeded for a longer period of time. In this case, the tamperer has done a good job if the undesired behavior appears randomly: he can just restart the program and hope that it will work next time. However, if it is linked to certain input patterns or hardware identifiers, the usability of the tampered version is decreased significantly.

4 Tools of the Trade

The core mechanism behind the discussed techniques is illustrated in Figure 1. In its simplest incarnation, a piece of code $c \in C$ is duplicated, both copies are diversified and one of them is selected at run time more or less randomly. Note that C represents the set of syntactically correct pieces of code in whatever language it is written.

This section provides more detail on two aspects related to the input of the opaque predicates. Firstly, we present techniques to augment the user-observed input with chaff input as a source of randomness. Secondly, we discuss the usage of variable program state at a program point – e.g., as fake input dependencies. Finally, we also look into more detail on how to generate diverse copies of a piece of code, as we believe that this has not been discussed in sufficient detail in other publications. Due to the extent of the discussion, it is in a separate section (5). Other aspects, such as the creation of the opaque predicates and the rewriting of software, have been discussed elaborately elsewhere.

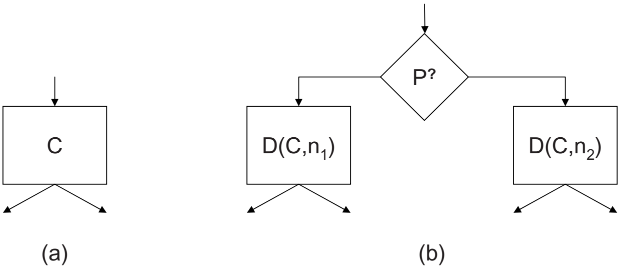


Fig. 1. The basic mechanism behind the implementation

4.1 Chaff Input

We say that run-time randomization delays the locate phase if it introduces diversity during a single "tamper session" – i.e., if the randomization takes place even on a single computer, for the same user-observed input, and for a limited period of time (one or a few days). Conversely, run-time randomization is said to delay the test phase if it requires multiple tamper sessions – i.e., the tampering itself needs to be repeated for different computers, for different user-observed inputs, and for different periods of time.

Under these specifications, chaff input is needed to delay the locate phase. This will be used as a source of pseudo-randomness, which will then serve as input to the opaque predicates. Note that chaff input is likely to stand out in command-line applications, as there is typically little difference between the user-observed and the fully specified input. However, real-life interactive applications already make use of threads, timing information, information about mouse movement, on-line content, etc., making them more suited for this technique.

We now discuss some sources of chaff input.

- **Scheduling of Threads.** In multi-threaded applications, several threads may interact with one another in a non-deterministic manner (from the viewpoint of the application). The actual scheduling depends on the operating system (and the virtual machine, if applicable), and is influenced by asynchronous events such as user interaction, other processes, thread priority, etc. Therefore, the actual scheduling is an excellent source of pseudo-randomness. If necessary, additional threads can be created to perform part of the original functionality, or to perform other software-protection tasks.
- **Return Values of System Calls.** System calls also provide a source of randomness from the viewpoint of the application. Many system (or library) calls return information that is changeable over different runs: system time, unallocated memory, network traffic, load of the machine, file system, etc.
  The Underhanded C Code Contest 2005 (www.brainhz.com/underhanded/) contains examples on how to obtain pseudo-randomness in a covert way. One of the entries leaves a matrix partially uninitialized, as a result of which it still contains information from a previous `stat()`-call (`stat` returns file

info, including time of last access). This type of call is common in regular programs and will thus not quickly raise suspicion.

An interesting way of randomizing the program is to change the code executed (not the behavior) based on the presence of a debugger. This way an attacker could spend much time making the program behave as desired in the debugger, only to find that it behaves differently without the debugger.

– **External Service.** Alternatively, we may require access to an external service, which provides a source of randomness. Such an external service could be a piece of trusted hardware or an on-line service.

Record/replay mechanisms and omniscient debuggers. Clearly, given a fully specified input, the behavior will be deterministic. While the fully specified input is often a superset of what the user perceives as input, a tamperer could ultimately use a perfect record/replay system [22] to make the fully specified input (including data, user interaction, communication, system calls, schedules, etc. [26]) repeatable, thus making the execution repeatable. This way he can track down and tamper with one of the copies of the origins of the undesired behavior. Alternatively, he could use an omniscient debugger [5], such as the Simics Hindsight Debugger, to back-track to the origin. Note that the general application of these techniques can be very expensive in terms of memory requirements. Therefore, a potential defense against such capabilities is to increase the amount of state necessary for the debugger to be able to trace backwards. This can be accomplished by maximizing the number of irreversible operations in the program.

In any case, there will be more origins of the undesired behavior. Unless the tamperer finds a way to automate detecting copies of that specific origin, which is undecidable in general, he must either (i) repeat this labor-intensive method for every copy of the origin, or (ii) make the choice between the different copies fixed. As a result, the workload of the tamperer increases.

Fixing the choice between different copies may be complicated as well. It may be easy automatically to find points where the different executions digress, but some of these points may be part of the original functionality of the program.

4.2 Variable Program State and Fake Input Dependencies

The internal state at a program point is itself highly variable, and therefore serves as an excellent source of input for the opaque variables. Furthermore, it is less suspicious to select different execution paths based upon the internal state.

Through profiling [20], we can easily spot tuples (p, s), for which either (i) the state s is constant at program point p for a fixed input, but variable for different inputs; or (ii) the state s is variable at program point p even for a fixed input. Note that, due to the nature of profiling, we cannot be certain that a state s is fixed; we can conclude only that a state s is fixed for the tested inputs.

Tuples for which the first property holds are then candidates for introducing fake input dependencies. As a result, execution for different inputs will differ at places where it originally overlapped. This can thus delay the test phase.

Tuples for which the second property holds are useful to delay the locate phase, because they will increase the amount of information in the static representation of the program and the number of different instructions in a trace of a particular execution. As a result, the trace will be less "foldable," by which we mean that constructing a Control-Flow Graph (CFG) from the trace will result in a larger CFG than from the original program.

Using the same argument as earlier, an attacker needs to patch at least one of the copies, and needs to patch additional ones or remove the fake dependencies on the program state. The latter may be harder than in the earlier case. These kinds of choices during execution of the program are bound to be less suspicious, since this type of choice occurs regularly during normal operation.

5 Diversity Systems

Diversity (also referred to as individualization or randomization) can be applied in a number of different ways. The most heard-of form of randomization is probably Address Space Layout Randomization (ASLR). ASLR is a specific form of randomization that requires no changes to the program itself. Instead, the operating system positions key code and data areas in a random way to make it harder to predict target addresses.

This type of defense is less viable in the malicious-host model, since we cannot rely on the environment. The only aspect that we can control is the program itself; thus, the randomization needs to be an integral part of it. Under these circumstances, there are still a number of possibilities on when and where to randomize:

- **Before distribution:** The static representation of the program is randomized before distribution.
- **During installation:** The static representation of the program is randomized when it is installed (e.g., based upon hardware, the license key, etc.)
- **Between runs:** The static representation of the program is randomized between executions, comparable to metamorphic viruses.
- **During execution:** The dynamic execution trace of the program is randomized, as discussed in this paper.

All of these types have in common that some system is needed to generate semantically equivalent, but syntactically different versions of a piece of code.

A schematic diagram of such a "diversity system" is given in Figure 2. A diversifier D takes as input a piece of code c and a set of nonces (numbers used once) $\{1, \ldots, k\}$, and produces a set of code pieces $\{D(c, 1), \ldots, D(c,_k)\}$ so that $\forall i \in [1, k] : D(c, i)$ has the same functionality as c, yet $\forall (i, j) \in [1, k]^2$, $D(c, i)$ is syntactically different from $D(c, j)$.

Similar to Kerckhoffs' principle for cryptography (a cryptosystem should be secure even if everything about the system, except the key, is public knowledge), we must assume that everything about the system is public knowledge. We

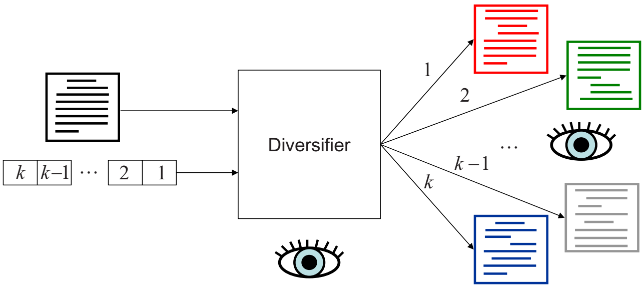


Fig. 2. Schematic of a diversity scheme

should also assume that an attacker will have access to one or more of the diversified versions. Note that this is obvious in the case of run-time randomization, as everything is embedded in the program.

5.1 Combining Diversity Systems

Rather than trying to build a single monolithic diversity system from scratch, we chose to build upon the vast body of existing research on semantics-preserving program transformations. Semantics-preserving transformations have been developed for a wide range of applications, such as refactoring, optimization for size and speed, obfuscation, watermarking, instrumentation, etc.

Combining many small transformations also makes it easier to prove (or debug) their semantics-preserving nature independently. If every transformation is semantics-preserving, the combination is guaranteed to be semantics-preserving.

The goal of combining these transformations is to enlarge the set of semantically equivalent pieces of code that can be generated by the resulting diversity system D from a piece of code $c \in C$. As usual, this is referred to as the "range" property, $ran(D, c)$.

The cardinality of the range of a diversity system can easily become very large. Consider the diversity system which chooses for every instruction in the original code whether or not to precede it by a nop-instruction. If that piece of code consists of n instructions, then the cardinality of the range is 2^n. This diversity system has a big range, yet can be easily circumvented in most applications of diversity. Therefore, the cardinality of the range is not a good indication of the quality of a diversity system.

On the other hand, a diversifier E of which the range is a superset of the range of another diversifier D ($\forall c \in C : ran(D, c) \subseteq ran(E, c)$) will typically be preferred, as this indicates that more diversity can be achieved. We will abbreviate this relation as follows: $D \subseteq E$.

Choice operation. Fortunately, given two diversity systems D and E, it is easy to create a third diversifier F for which $ran(D) \subseteq ran(F)$ and $ran(E) \subseteq ran(F)$ through the choice operation (Figure 3a): $F = D \vee E$. This corresponds to making

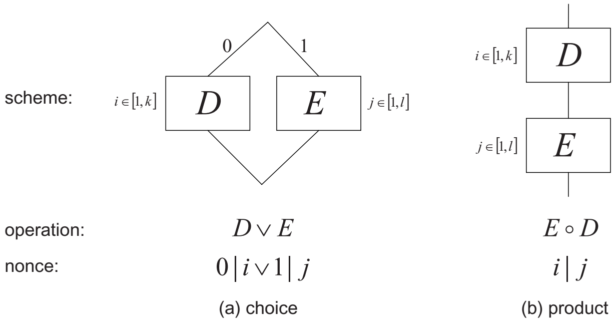


Fig. 3. Two combining operations for diversity schemes

a preliminary choice as to whether system D or E is to be used. When this is done, D or E is used as originally defined. Note that $D \vee E = E \vee D$.

Product operation. A second combining operation (Figure 3b): $F = E \circ D$, corresponds to diversifying the program with the first diversifier D and diversifying the resulting program with the second diversifier E, the nonces for D and E being chosen independently. This total operation is a diversifier whose transformations consist of all the products (in the usual sense of products of transformations) of transformations in E with transformations in D.

Logging the applied transformations. Keeping track of the applied transformations is important, since some applications may need to be able to recreate the diversified copies. For example, consider randomizing programs before distribution. When updates are needed afterwards, the software provider may need to tailor them to a specific copy. Maintaining a database of nonces requires less storage than keeping a copy of every distributed version.

This choice operation is recorded in the resulting nonce as follows: If $[1, k]$ (respectively $[1, l]$) is the range of nonces accepted by D (E respectively), then for $i \in [1, k], j \in [1, l]$, the nonce becomes $0|i \vee 1|j$. ('|' is the concatenation operator). The nonce of the product operation then becomes $i|j$.

5.2 Injective Properties of a Diversity System

Nonce-injective. An important property of a diversity system is that the resulting programs be in fact diverse. Therefore, we want different nonces to lead to different programs – i.e.,

$$\forall c \in C, \forall i, j \in [1, k], i \neq j : D(c, i) \neq D(c, j)$$

Note that the relation $=: P \times P$ denotes syntactical equivalence. We call this property *nonce-injective* – i.e., $\forall c \in C, D(c, .)$ is injective.

Typically, this will not be a problem for basic transformations (not a composition of other transformations). However, it may become an issue when many transformations are combined using the combination operations described earlier, since the product of two nonce-injective transformations is not necessarily nonce-injective.

Injective. A transformation is injective in the traditional sense if:

$$\forall(c_1, c_2) \in C^2, \forall(i,j) \in [1,k]^2, c_1 \neq c_2 \vee i \neq j : D(c_1, i) \neq D(c_2, j)$$

Clearly, the composition of two injective transformations is injective. Furthermore, if E is an injective transformation and D is a nonce-injective transformation, then $E \circ D$ is nonce-injective. Note that if c_1 and c_2 have different semantics, c_1 cannot be syntactically equal to c_2. This definition is useful when c_1 and c_1 are two semantically equivalent, but syntactically different versions of a piece of code (e.g., after applying a diversity system).

Disjoint diversity systems. We say that two diversity systems D and E are *"disjoint"* if and only if

$$\forall c \in C, \forall i \in [1,k], \nexists j \in [1,l] : D(c,i) = E(c,j)$$

The choice of two disjoint injective transformations $(D \vee E)$ is injective.

5.3 Diversity Systems in Practice

A practical diversity system may be composed of a number of transformations: $(D_1 \vee D_2 \vee ... \vee D_n) \circ (D_1 \vee D_2 \vee ... \vee D_n) \circ (D_1 \vee D_2 \vee ... \vee D_n)$.... The probabilities that determine which transformation to choose in every iteration are assignable, and may change as the result of earlier transformations. For example, it may be useless to apply the same transformation twice. This can be recorded by setting its probability to zero for subsequent iterations once it is selected. For a more elaborate discussion on the selection of transformations with dependencies, we refer to closely related work on selecting transformations in the domain of obfuscation by Heffner and Collberg [16].

Iterated transformations. To increase the range and complexity of randomization, a tool can iterate and recombine a number of diversifying operators. Each such primitive can be quite simple – e.g., referencing variables through newly created pointers or duplicating a program statement, along with a new obfuscated predicate to choose one of the individualized copies. While such operators may be insecure when used alone, iterated application can create complexity, including emergent properties due to interaction among various transformations. This is similar to behavior found in complex systems such as cellular automata, and also helps to create confusion and diffusion, as in iterated application of rounds in block ciphers and hash functions.

Selecting transformations for the composed diversity system. The injective property and related properties discussed earlier prove to be a useful guideline in the selection of transformations to add to the mix. For example, it is not useful to add a transformation D to a diversity system E if the range is not increased as a result ($D \subseteq E$).

Clearly, injective transformations disjoint with the already present diversity system are preferred. In practice, however, this requirement is not so stringent: Due to the large range, the chance of actually obtaining two identical code pieces after a number of transformations is small. If required, a hash can be computed of every generated code piece, and newly generated code pieces can simply be discarded if their hash matches one of the earlier ones.

Selecting nonces. In practice, it proves to be complicated to determine the range of nonces accepted by a composed diversity system. The application of one transformation will lead to more or fewer possibilities for the next transformation in a way that is hard to predict without actually applying the transformation. As the range quickly becomes unmanageable, generating all possibilities to determine the range in advance is also not practically viable. Therefore, we cannot predetermine a uniform range of nonces from which to choose in advance. Rather, every transformation will return its range once it is selected as the next transformation (and all previous transformations have been applied), after which an element from its range is selected. The nonces are thus built dynamically during the randomization, as shown in Figure 3, and can have variable lengths.

6 Evaluation

To get an idea of the achievable range of practical diversity systems, we have implemented a number of diversifying transformations in the binary rewriting framework Diablo [11]:

1. Splitting basic blocks by a two-way opaque predicate (as shown in Figure 1).
2. Inlining basic blocks with multiple incoming edges (as shown in Figure 4).
3. Inlining functions.
4. Replacing instructions by semantically equivalent instructions.
5. Reordering instructions within a basic block.
6. Inverting the condition of branches.
7. Reordering chains of basic blocks.

We have evaluated these transformatiosn on the C programs of the SPEC 2006 benchmark suite, compiled with gcc 3.2.2 and statically linked against glibc 3.2.2. The number of choices that need to be made for the transformations when applied to the entire benchmark is given in Table 1, normalized to choices between 10 options. For perlbmk, e.g., we can choose independently for $83,759$ basic blocks whether or not to split them with a two-way opaque predicate. This leads to $2^{83,759}$ possible output programs (assuming we use the same predicate every time), or a range of about $10^{25,214}$; hence the value $25,214$ in the table. The second transformation has been limited to one round, meaning that the candidates for inlining, namely (basic-block, edge) pairs, are all taken from the

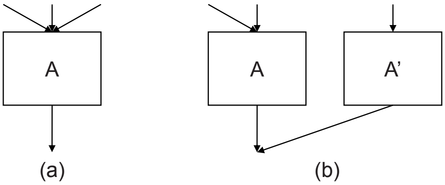


Fig. 4. Inlining basic blocks with multiple incoming edges

Table 1. Number of available choices x, normalized to 10 options. The range is 10^x.

	400.perlbench	401.bzip2	403.gcc	429.mcf	433.milc	445.gobmk	456.hmmer	458,sjeng	462.libquantum	464.h264ref	470.lbm	482.sphinx
1. split bbl	25214	6704	63948	6093	7289	16519	8355	7679	6380	12150	6146	8066
2. inline bbl	16434	3908	46098	3541	4313	10384	5037	4561	3732	6947	3585	4834
3. inline fun	2734	594	6800	508	803	2254	944	697	545	1039	513	999
4. select ins	20630	7340	51413	6603	7464	16203	8495	8251	6876	13210	7301	8228
5. schedule	14330	5679	31032	4741	5972	15462	6518	5985	5161	13617	4813	6450
6. flip branch	11516	3106	29803	2781	3184	7171	3676	3540	2888	5547	2803	3572
7. layout	76379	19319	183010	17954	21160	51065	23913	22655	18646	34457	18159	22777

original control-flow graph. If more rounds are allowed, the transformation can be reapplied endlessly for constructs such as loops (a loop can be unrolled infinitely).

The bars in the table aim intuitively to indicate the per-benchmark relation among the available choices for the different transformations.

In order to evaluate the cost of the techniques discussed in this paper, we have evaluated the impact on the code size and execution time resulting from the following setup: Transformations 1-3 are applied with a probability drawn from a Bernoulli distribution with $p = 0.05$. As a result, the transformations are applied about 5% of the times they could be applied. Both the original and copied version are then diversified by randomly applying transformations 4-7.

Over the entire benchmark suite, we notice an increase of about a quarter in code size. This is slightly higher than what one may expect at first (about 15,8% from three times 5% increase), because the candidates for transformation 2 are (basic-block, edge) pairs, which is more than just basic blocks. The same holds for transformation 3 where the candidates are (function, call-site) pairs.

The slowdown is on average 7%. This slowdown results from (i) the evaluation of the opaque predicates, (ii) additional control-flow instructions, and (iii) worse cache behavior due to less code locality and increased code size. Note that the slowdown is more variable than the code-size increase, as it depends on the execution count of the transformed code.

7 Related Work

Software Diversity. Software diversity was first used for *fault tolerance* as an extension of the idea of using redundant hardware to mitigate physical faults.

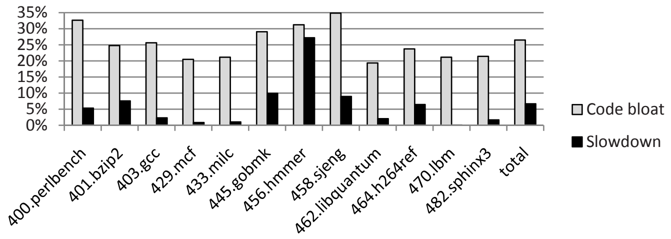


Fig. 5. Code bloat and slowdown for an exemplary transformation

The two main directions are recovery-block software [21] and N-version programming [2]. Both rely on hand-written, semantically equivalent modules. Recovery-block software requires an acceptance test, and the implementation with the highest priority to pass the test wins. N-version programming compares the outputs produced by several versions and propagates only consensus results.

Software diversity or individualization as a security mechanism *against malicious code* attacks was proposed by Cohen [8] under the term "program evolution." Since then, numerous transformation techniques have been presented, including memory-layout randomization [13] and instruction-set randomization [4]. It has been shown that these techniques are also vulnerable to attacks [23,24].

Other research assumes the presence of diversity and studies the assignment of distinct software packages to individual systems in a network [18], or uses different versions in a framework for detection and disruption of attacks, similar to N-version programming for fault tolerance [10].

Software diversity as a protection mechanism *against a malicious host* seems to have received less attention. Existing work is focused on randomization before distribution. Anckaert et al. [1] propose to rewrite the program in a custom instruction set and to ship it with a matching virtual machine. Zhou et al. [27] present code transformations based upon algebraic structures compatible with 32-bit operations commonly present in code.

Software diversity has also been used to *hide malicious code*, such as viruses. Self-modifying viruses will typically change their binary representation before propagation. Early implementations simply encrypt the body of the virus with a different key, leaving the decryption routine vulnerable to signature-based detection. More recent viruses diversify the decryption routine as well, or contain a metamorphic engine to rewrite themselves completely (e.g., W32.simile[1]).

Tamper-Resistance. Most techniques to protect the integrity of a program are based on checksumming segments of the code [6,17]. A generic attack against such schemes has been devised for the x86 through the manipulation of processor-level segments, and for the UltraSparc through a special translation look-aside buffer load mechanism [25]. A countermeasure against this type of attack relies on self-modifying code [15]. Related techniques [7] hash the execution of a piece

[1] http://securityresponse.symantec.com/avcenter/venc/data/w32.simile.html

of code, while others have looked at the reaction mechanism in more detail. Once tampering is detected, appropriate action needs to be taken. If the manifestation of this action is too obvious, it can be easily tracked down. Delayed and controlled failures [14] are a way to make it harder to locate the reaction mechanism.

Obfuscation. Software obfuscation [3,9] aims to make programs harder to understand and has many parallels with software diversity. While the goals are different, many of the techniques developed for obfuscation can be parameterized for diversity purposes. Typically, the versions of a piece of code generated by a diversity system will be obfuscated. If the different versions are too easy to understand, it may be easy to match or find semantically equivalent code.

8 Conclusion

We modeled the tamperer's behavior starting from parallels between debugging and tampering. As such, the techniques presented to mitigate tampering leverage known difficulties from debugging: non-deterministic behavior and the fundamental limitations of testing. An experimental evaluation shows that diversity systems can generate many different semantically equivalent code sequences and that the cost of the applied techniques is acceptable for most applications.

References

1. Anckaert, B., Jakubowski, M., Venkatesan, R.: Proteus: virtualization for diversified tamper-resistance. In: The 6th ACM workshop on Digital Rights Management, pp. 47–58 (2006)
2. Avizienis, A., Chen, L.: On the implementation of N-version programming for software fault tolerance during execution. In: The 1st IEEE Computer Software and Applications Conference, pp. 149–155 (1977)
3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
4. Barrantes, E., Ackley, D., Forrest, S., Stefanovi, D.: Randomized instruction set emulation. ACM Trans. on Information and System Security 8(1), 3–40 (2005)
5. Bhansali, S., Chen, W., de Jong, S., Edwards, A., Murray, R., Drinic, M., Mihocka, D., Chau, J.: Framework for instruction-level tracing and analysis of program executions. In: Virtual Execution Environments Conference (2006)
6. Chang, H., Atallah, M.: Protecting software code by guards. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, pp. 160–175. Springer, Heidelberg (2002)
7. Chen, Y., Venkatesan, R., Cary, M., Pang, R., Sinha, S., Jakubowski, M.: Oblivious hashing: a stealthy software integrity verification primitive. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 400–414. Springer, Heidelberg (2003)
8. Cohen, F.: Operating system evolution through program evolution. Computers and Security 12(6), 565–584 (1993)
9. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: The 25th Conference on Principles of Programming Languages, pp. 184–196 (1998)

10. Cox, B., Evans, D., Filipi, A., Rowanhill, J., Hu, W., Davidson, J., Knight, J., Nguyen-Tuong, A., Hiser, J.: N-variant systems: A secretless framework for security through diversity. In: The 15th USENIX Security Symposium, pp. 105–120 (2006)
11. De Sutter, B., De Bus, B., De Bosschere, K.: Link-time binary rewriting techniques for program compaction. ACM Trans. on Programming Languages and Systems 27(5), 882–945 (2005)
12. DiMarzio, J.F.: The Debugger's Handbook. Auerbach Publications (2007)
13. Forrest, S., Somayaji, A., Ackley, D.: Building diverse computer systems. In: The Workshop on Hot Topics in Operating Systems, pp. 67–72 (1997)
14. Gang, T., Yuqun, C., Jakubowski, M.: Delayed and controlled failures in tamper-resistant systems. In: The 8th Information Hiding Conference (2006)
15. Giffin, J., Christodorescu, M., Kruger, L.: Strengthening software self-checksumming via self-modifying code. In: The 21st Annual Computer Security Applications Conference, pp. 23–32 (2005)
16. Heffner, K., Collberg, C.: The obfuscation executive. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 428–440. Springer, Heidelberg (2004)
17. Horne, B., Matheson, L., Sheehan, C., Tarjan, R.: Dynamic self-checking techniques for improved tamper resistance. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, pp. 141–159. Springer, Heidelberg (2002)
18. O'Donnell, A., Sethu, H.: On achieving software diversity for improved network security using distributed coloring algorithms. In: The 11th ACM conference on Computer and Communications Security, pp. 121–131 (2004)
19. Park, Y., Lee, G.: Repairing return address stack for buffer overflow protection. In: The 1st conference on Computing frontiers, pp. 335–342 (2004)
20. Pettis, K., Hansen, R.: Profile guided code positioning. In: The ACM conference on Programming Language Design and Implementation, pp. 16–27 (1990)
21. Randell, B.: System structure for software fault tolerance. SIGPLAN Notices 10(6), 437–449 (1975)
22. Ronsse, M., De Bosschere, K.: Recplay: a fully integrated practical record/replay system. ACM Transactions Computer Systems 17(2), 133–152 (1999)
23. Shacham, H., Page, M., Pfaff, B., Goh, E., Modadugu, N., Boneh, D.: On the effectiveness of address-space randomization. In: The 11th ACM conference on Computer and communications security, pp. 298–307 (2004)
24. Sovarel, A., Evans, D., Paul, N.: Where is the FEEB? The effectiveness of instruction set randomization. In: The 14th USENIX Security Symposium, pp. 145–160 (2005)
25. Wurster, G., van Oorschot, P., Somayaji, A.: A generic attack on checksumming-based software tamper resistance. In: The 26th IEEE Symposium on Security and Privacy, pp. 127–138 (2005)
26. Zeller, A.: Why Programs Fail: A Guide to Systematic Debugging. Morgan Kaufmann, San Francisco (2005)
27. Zhou, Y., Main, A.: Diversity via code transformations: A solution for NGNA renewable security. In: NCTA - The National Show (2006)

Privacy-Preserving Eigentaste-Based Collaborative Filtering

Ibrahim Yakut and Huseyin Polat

Computer Engineering Department, Anadolu University, Eskisehir, 26470 Turkey
{iyakut, polath}@anadolu.edu.tr

Abstract. With the evolution of e-commerce, privacy is becoming a major concern. Many e-companies employ collaborative filtering (CF) techniques to increase their sales by providing truthful recommendations to customers. Many algorithms have been employed for CF purposes; and Eigentaste-based algorithm is one of them. Customers' preferences about products they purchased previously or showed interest are needed to provide recommendations. However, due to privacy concerns, customers refuse to contribute their ratings at all; or they might decide to give false data. Providing truthful referrals based on such inadequate and false data is impossible. Therefore, providing privacy measures is vital for collecting truthful data and producing recommendations.

In this paper, we investigate how to achieve CF tasks (predictions and top-N recommendations) using Eigentaste, which is a constant time CF algorithm, without greatly exposing users' privacy. To accomplish privacy, we employ randomized perturbation techniques (RPT). We modify and/or simplify original Eigentaste algorithm in such a way to provide private referrals efficiently with decent accuracy. We investigate our proposed schemes in terms of privacy. To evaluate the overall performance of our schemes, we conduct experiments using real data sets. We then analyze our outcomes and finally provide some suggestions.

1 Introduction

CF techniques are widely used by many e-companies for recommendation purposes. With the number of users accessing the Internet and the products available online growing, CF techniques are becoming increasingly popular as part of online shopping sites. These sites incorporate recommendation systems that suggest products to customers based on items that like-minded users have ordered before, or have indicated as being of interest. CF has many important applications in E-commerce, direct recommendations, and search engines [4,5]. With the help of CF, users can get referrals about many of their interests and activities; including, but not limited to restaurants, bars, movies, books, news, music CDs, and interesting things to do in a city.

To provide accurate referrals efficiently in terms of computational complexity, many algorithms have been proposed. Since accuracy and efficiency are conflicting, the goal might become to find a balance between them. Eigentaste [10] is a

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 169–184, 2007.
© Springer-Verlag Berlin Heidelberg 2007

CF algorithm that uses universal queries on a common set of items and applies principal component analysis (PCA). It requires constant time to compute predictions, given a database of n users. Some of the computations in Eigentaste can be done off-line, while recommendation computations are done online. Recursive rectangular clustering (RRC) is employed to cluster users. CF systems perform two types of tasks [19]: Providing prediction for a single item (target item, q) showing how much a customer (an active user, a) will like q and producing a sorted list of items (top-N recommendations) that will be liked by a.

There is a great potential for individuals to share all kinds of information about places and things to do, see and buy; but the privacy risks are many and severe. E-commerce personalization poses various privacy risks [6]. The first one is unsolicited marketing. Secondly, many users are concerned about the use of computers to pry into their personal lives. People are also worried that vendors will profile them to facilitate price discrimination. Another risk is that data in users' profiles might be used in a criminal case. And finally, data collected for CF purposes might be used for government surveillance. Moreover, customer data is a valuable asset, and it has been sold when some e-companies suffered bankruptcy [5]. Although some people might be willing to selectively divulge information, a significant number of people are not willing to divulge their information, due to privacy concerns, according to a survey conducted in 1999 [7]. Users also dislike data transfer and data being shared with other companies. Due to such concerns, it becomes a challenge to collect data, especially truthful and trustworthy data, for CF purposes. Providing privacy measures plays a vital role to collect truthful data and to produce accurate referrals. *How can customers give their data for CF purposes without jeopardizing their privacy? Is it still possible to generate recommendations based on perturbed data?*

We investigate how to achieve CF tasks while preserving users' privacy. We study how to disguise customers' ratings to produce recommendations based on disguised ratings using Eigentaste algorithm with privacy. The main merit of Eigentaste algorithm is that it is a linear time algorithm and different kinds of clustering algorithms can be used with it. Our goal is to provide referrals with privacy via an algorithm that can utilize various types of clustering algorithms in linear time. CF algorithms should be accurate and efficient [10]. Moreover, when privacy is an issue, they should preserve users' privacy, as well. On the other hand, privacy, accuracy, and efficiency are conflicting goals. Increasing one or two of them decreases the other (s). Therefore, we wish for proposing solutions to find an equilibrium among them while achieving privacy-preserving Eigentaste-based CF. We analyze the proposed schemes in terms accuracy, privacy, and efficiency. We conduct experiments to assess the overall performance of our proposed schemes. We analyze the outcomes of the experiments.

2 Related Work

Privacy-preserving collaborative filtering (PPCF) is becoming increasingly popular with the evolution of e-commerce. Canny proposes alternative models for

PPCF allowing users to control their data [4,5]. A community of users can compute a public "aggregate" of their data without disclosing any individual user's data. The aggregate allows personalized referrals to be computed by members of the community, or by outsiders. Homomorphic encryption is employed to allow sums of encrypted vectors to be computed and decrypted without exposing individual data. Unlike his schemes, in our schemes, users do not actively participate in CF process after they contribute their data to the server. His schemes are based on distributed computations, while ours are based on central computations. We employ randomized perturbation techniques (RPT) for data disguising, while privacy achieved using cryptographic techniques in his schemes.

Polat and Du employ the RPT for PPCF [14]. They have shown that it is possible to provide accurate predictions while preserving users' privacy. They employ neighborhood- and singular value decomposition (SVD)-based algorithms. However, we discuss Eigentaste-based CF with privacy, where we investigate providing clustering based-CF services on disguised data. In [18], providing predictions based on inconsistently disguised data using memory- or correlation-based algorithms is discussed. However, in this paper, we show how to achieve private CF services (both predictions and top-N recommendations) using Eigentaste-based algorithms. In [17], achieving referrals using item-based algorithms on binary ratings without unduly eroding users' privacy is discussed. Randomized response techniques (RRT) is employed to perturb users' data while still producing accurate predictions. Although they use item-based algorithms and the RRT, we use Eigentaste-based algorithms and the RPT. In our work, we employ the RPT, which seems proper for non-binary ratings in contrast to the RRT, which is just feasible for binary ratings.

In [15,16], providing private referrals on partitioned data is discussed. Two vendors want to conduct CF on the integrated data without revealing their private data to each other. They exchange some aggregate data off-line and online while preserving their privacy. Unlike such schemes, where data is partitioned between different parties, our scheme is based on existing databases. The server collects perturbed data from many users and creates a database containing masked data. It then performs CF tasks based on such existing database.

3 Background: Principal Component Analysis (PCA) and Eigentaste-Based CF

Principal component analysis (PCA) is a closely-related factor analysis technique and reduces dimensionality by optimally projecting highly correlated data along a smaller number of orthogonal dimensions [10]. PCA facilitates dimensionality reduction for off-line clustering of users and rapid computation of referrals. Goldberg et al. [10] apply eigen-analysis to solve for matrices E and Λ such that

$$C = E^T \Lambda E \tag{1}$$

and

$$ECE^T = \Lambda, \tag{2}$$

where C is correlation matrix, E is orthogonal matrix of eigenvectors of C, and Λ is diagonal matrix of eigenvalues of C. After finding eigenvectors, they keep principal eigenvectors only. The number of eigenvectors to retain depends on eigenvalues and it is small. If v eigenvectors are retained, data is projected along the first v principal eigenvectors:

$$x = AE_v^T, \tag{3}$$

where A is normalized matrix of users ratings of items in gauge set. The popular choice is to set v at 2, so that data are projected onto eigen-plane.

Eigentaste algorithm can be described, as follows: Given an $n \times m$ matrix of raw ratings from n users and m items, Goldberg et al. [10] select k of these items to form a gauge set. They normalize the gauge set to produce A, which is an $n \times k$ matrix. Each rating is normalized by subtracting its mean rating over all users, and then dividing by its standard deviation. They define the global correlation matrix C over all users:

$$C = \frac{1}{n-1} A^T A, \tag{4}$$

where C is symmetric and positive definite. After projecting data onto eigen-plane, users can be clustered. They implement a recursive rectangular clustering (RRC) to cluster users. Each cell is treated as a cluster of neighbors in the eigen-plane. For each cluster, the mean for each non-gauge item is computed based on the number of users who rated that item. Sorting the non-gauge items in order of decreasing mean ratings yields a lookup table of recommendations for that cluster. The computations so far are done off-line.

To compute recommendations online, a new user (an active user, a) contribute his/her ratings for all items in the gauge set. Using principal components, the data entered is projected onto the eigen-plane. After finding representative cluster, recommendations are presented with the help of lookup table.

4 Privacy-Preserving Eigentaste-Based CF

Our goal is to provide private referrals efficiently with decent accuracy. Efficiency can be explained, as follows: Additional online costs like storage, communication, and computation costs should be small and negligible because off-line costs are not critical to overall performance. We can define accuracy, as follows: Predictions computed based on perturbed data should be close to true rating values. And finally, although it is a challenge to define privacy clearly, we can explain privacy in the context of CF, as follows: The server or the data collector should not be able to learn the true ratings and the rated items of users including active users. Users do not want to reveal their true ratings about products they bought or showed interest. Moreover, it might be more damaging to disclose the purchased products than revealing the true votes. For example, nobody wants to reveal that he/she bought pornographic magazine or visited pornographic sites. Therefore, besides perturbing true ratings, users disguise unrated items' cells, as well. Since privacy, accuracy, and efficiency are conflicting goals, we aim to find a good balance between them.

4.1 Modified Eigentaste-Based CF Algorithm

We modified original Eigentaste algorithm in such a way to achieve our goals, as follows: To normalize ratings into z-scores, we use user-mean votes rather than item-mean ratings. Each user is able to compute z-scores of his/her ratings without the help of other users if user-mean votes are used for normalization, as follows:

$$z_{uj} = \frac{v_{uj} - \overline{v_u}}{\sigma_u}, \tag{5}$$

where v_{uj} is the true rating of user u on item j, $\overline{v_u}$ is the mean rating of user u, σ_u is the standard deviation of user u's ratings, and z_{uj} is the z-score value of user u on item j.

Besides employing the RRC as done in Eigentaste algorithm, there are alternative clustering algorithms like k-means, fuzzy-C means, etc. It is possible to apply different clustering algorithms to achieve privacy while providing accurate referrals efficiently. We propose to use k-means clustering algorithm to cluster the projected data into 57 clusters because recent results show that PCA components provide solutions to k-means clustering [8,13].

In Eigentaste, to create the lookup table, non-gauge items' means are used. We propose to use the z-score values' means to create the lookup table. Once the cluster of a is found, the mean value of z-scores of the target item (q) is provided to a. Then, a can de-normalize it and finds the prediction (p_{aq}) for q, as follows:

$$p_{aq} = \overline{v_a} + \sigma_a \times \overline{z_q}, \tag{6}$$

where $\overline{v_a}$ is the mean rating of a's ratings, σ_a is the standard deviation of a's ratings, and $\overline{z_q}$ is the mean z-score value of q.

In addition to employing lookup table, alternative methods can be applied to calculate the referrals online. Using lookup table is advantageous due to small computation time. On the other hand, it might be possible to increase accuracy while sacrificing on computation complexity. It is possible to use well-known memory-based CF algorithms to compute the referrals online. After finding a's representative cluster, we can find predictions by computing the weighted sum of co-rated items between a and users in that cluster.

4.2 Data Masking

There are various methods to mask private data. One method is to employ randomized perturbation techniques (RPT). To disguise a number x, a simple way is to add a random value r to it. $x+r$, rather than x, will appear in the database, where r is a random value drawn from some distribution with mean (μ) being 0. The basic idea of the RPT is to perturb the data in such a way that the server can only know the range of the data, and such range is broad enough to preserve users' privacy. Although information from each individual user is scrambled, if the number of users and/or items are significantly large, the aggregate information of these users can be estimated with decent accuracy. Such a property is useful for computations that are based on aggregate information. For those computations, we can still generate meaningful outcomes without knowing the exact values of individual data items because the required aggregate information can be estimated from scrambled data. Since Eigentaste-based CF is also based on aggregate information rather than individual data items, the RPT can be applied to them. Therefore, we integrate the RPT and Eigentaste CF algorithm to provide accurate recommendations with privacy.

After generating random data, users disguise their data by adding those random data to their corresponding ratings vector's cells. We can summarize the data masking process, as follows:

1. Each user u computes z-score values of their ratings.
2. Users and the server decide γ, θ, and δ values.
3. Each user u selects the standard deviation of the random numbers (σ_u) uniformly randomly over the range [0, γ]. Users generate the random numbers from a distribution with μ being 0 and σ_u. Each user u then uniformly randomly selects a random number r_u over the range [0, 1]. If $r_u \leq \theta$, uniform perturbing data; otherwise Gaussian perturbing data is used for data perturbation. Each user u finally uniformly randomly selects an integer x_{er} over the range [0, δ]. We define x_{er} as the percentage of unrated items' cells to be filled with noise data. They then randomly select the x_{er} percent of their unrated items' cells to be disguised.
4. Each user u creates m_u number of random numbers using uniform or Gaussian distribution with the selected σ_u values, where m_u is the number of noise data required to disguise the user u's private data including ratings and unrated cells. Such value depends on the number of ratings and the number of unrated items' cells to be disguised.
5. Users mask their private data by adding noise data to each of the cells to be perturbed. Finally, they send the disguised data to the server.

In uniform perturbing data, random values are generated using uniform distribution over the range $[-\alpha, \alpha]$, where α is a constant number and $\alpha = \sqrt{3}\sigma_u$. In Gaussian perturbing data, noise data is created using Gaussian distribution with σ_u. To obtain a balance between accuracy, privacy, and efficiency, γ, δ, and θ values can be adjusted.

4.3 Eigentaste-Based CF with Privacy

After collecting perturbed data from users, the server creates a disguised user-item matrix, D', which is an $n \times m$ matrix including disguised z-scores collected from n users for m items. k of those m items are selected to form the gauge set, A', which is an $n \times k$ matrix. Note that all users rated all items in gauge set. Therefore, A' is a dense set. The server then starts providing recommendations based on this perturbed matrix.

To provide recommendations, the server first estimates correlation matrix from perturbed data off-line, as follows:

$$C' = \frac{1}{n-1} A'^T A', \tag{7}$$

where C' is the estimated correlation matrix from perturbed data. We should show how the server can estimate C' from masked data. The entries other than the diagonal ones are computed, as follows:

$$\frac{1}{n-1}(A'^T A')_{fg} = \frac{1}{n-1}\sum_{u=1}^{n}(z_{uf}+r_{uf})(z_{ug}+r_{ug}) = \frac{1}{n-1}\sum_{u=1}^{n} z_{uf}z_{ug} + \frac{1}{n-1}\sum_{u=1}^{n} z_{uf}r_{ug} + \frac{1}{n-1}\sum_{u=1}^{n} z_{ug}r_{uf} + \frac{1}{n-1}\sum_{u=1}^{n} r_{uf}r_{ug} \approx \frac{1}{n-1}\sum_{u=1}^{n} z_{uf}z_{ug}, \tag{8}$$

where f and g show the row and column numbers, respectively, and $f \neq g$. Since random values r_{uf}s and r_{ug}s are independent and drawn from a distribution with $\mu = 0$, the expected value of $\frac{1}{n-1}\sum_{u=1}^{n} r_{uf}r_{ug}$ is 0. Similarly, the expected values of $\frac{1}{n-1}\sum_{u=1}^{n} z_{uf}r_{ug}$ and $\frac{1}{n-1}\sum_{u=1}^{n} z_{ug}r_{uf}$ are 0. However, since the scalar product is computed between the same vectors for the diagonal entries $(f = g)$, we can estimate them, as follows:

$$\frac{1}{n-1}(A'^T A')_{ff} = \frac{1}{n-1}\sum_{u=1}^{n}(z_{uf}+r_{uf})(z_{uf}+r_{uf}) = \frac{1}{n-1}\sum_{u=1}^{n} z_{uf}^2 + \frac{1}{n-1}2\sum_{u=1}^{n} z_{uf}r_{uf} + \frac{1}{n-1}\sum_{u=1}^{n} r_{uf}^2 \approx \frac{1}{n-1}\sum_{u=1}^{n} z_{uf}^2 + \frac{1}{n-1}\sum_{u=1}^{n} r_{uf}^2. \tag{9}$$

The expected value of $\frac{1}{n-1}\sum_{u=1}^{n} z_{uf}r_{uf}$ is 0 due to the same reasons. However, since we only need $\frac{1}{n-1}\sum_{u=1}^{n} z_{uf}^2$ values for diagonal entries, we need to get rid of $\frac{1}{n-1}\sum_{u=1}^{n} r_{uf}^2$ in Eq. (9), as follows, assuming $n \approx n-1$ for large n values:

$$\frac{1}{n-1}(A'^T A')_{ff} \approx \frac{1}{n-1}\sum_{u=1}^{n} z_{uf}^2 + \frac{1}{n-1}\sum_{u=1}^{n} r_{uf}^2 - \sigma_r^2 \approx \frac{1}{n-1}\sum_{u=1}^{n} z_{uf}^2, \tag{10}$$

where σ_r is the average standard deviation of random numbers. As explained before, with increasing n or in the long run, since random numbers are independently generated and drawn from distributions with μ being 0, the relative errors due to random numbers will converge to zero.

Since users disguise their private data using random numbers generated from some distribution with μ being 0 and σ_u, the server is able to estimate C' from perturbed data off-line, as explained previously. After estimating C', the server can apply eigen-analysis and estimate E', which is a $k \times k$ orthogonal matrix of eigenvectors of C'. It only keeps the v principal eigenvectors. v is usually small and generally set to 2. It then projects the data along the first $v = 2$ principal eigenvectors, as follows:

$$x' = A'E_2'^T, \tag{11}$$

where x' is an $n \times 2$ estimated matrix. The entries of x' can be estimated for all $i = 1, 2, \ldots, n$ and for all $j = 1, 2$, as follows, where e values represent the entries of E and R values represent the contributions of random values to true values of eigenvectors because they are estimated from perturbed data:

$$x'_{ij} = \sum_{l=1}^{k}(z_{il} + r_{il})(e_{lj} + R_{lj}) =$$
$$\sum_{l=1}^{k} z_{il}e_{lj} + \sum_{l=1}^{k} z_{il}R_{lj} + \sum_{l=1}^{k} r_{il}e_{lj} + \sum_{l=1}^{k} r_{il}R_{lj} \approx \sum_{l=1}^{k} z_{il}e_{lj}. \tag{12}$$

Due to the same reasons explained previously, the expected values of the last three summations in Eq. (12) is zero. Since summations are computed over k, with increasing number of gauge items (k), the relative errors due to random numbers will become smaller. Therefore, the server is able to project the disguised data along with the first two principal eigenvectors.

After projecting the masked data onto eigen-plane, the server clusters the projected data into various clusters. We prefer to use k-means clustering over the RRC because recent results show that PCA components provide solutions to k-means clustering [8,13]. The server finally generates lookup tables off-line. For each cluster, it finds the mean of the z-scores for each non-gauge items. Such average z-score values are stored in corresponding lookup tables.

The computations so far are conducted off-line. Once the server creates the model (generating clusters and lookup tables) off-line, it can start providing CF services to users. When an active user, a, wants recommendations, he/she sends his/her z-score values of items in gauge set together with a query to the server. We propose to use three different methods to preserve a's privacy and explained such schemes in the following subsection. The server can easily project a's data and find his/her corresponding cluster when a is used the first method to protect his/her privacy because a sends his/her true z-scores vector together with random vectors. When a employs the second or the third scheme to disguise the z-scores of items in the gauge set, the server first can project his/her z-scores, as follows, for all $j = 1, 2$:

$$x'_{aj} = \sum_{l=1}^{k}(z_{al} + r_{al})(e_{lj} + R_{lj}). \tag{13}$$

Since random numbers are drawn from some distribution with μ being 0, we can write the following approximation of Eq. (13):

$$x'_{aj} = \sum_{l=1}^{k}(z_{al} + r_{al})(e_{lj} + R_{lj}) \approx \sum_{l=1}^{k} z_{al}e_{lj}. \tag{14}$$

After projecting a's data, the server can now find the representative cluster. It finally sends the required average z-score for the target item q to a, who can de-normalize such value and estimates prediction for q.

4.4 Preserving Active Users' Privacy

We propose to use three methods to protect active users' privacy. The first method (M1) is based on the 1-out-of-n Oblivious Transfer protocol [3,9], which refers to a protocol where at the beginning of the protocol one party, Bob has n inputs $X_1, \ldots, X_n$ and at the end of the protocol the other party, Alice, learns one of the inputs X_i for some $1 \leq i \leq n$ of her choice, without learning anything about the other inputs and without allowing Bob to learn anything about i. By combining efficient protocols, the 1-out-of-n Oblivious Transfer protocol could be achieved with poly-logarithmic (in n) communication complexity. When a uses this method, he/she sends Y-1 randomly generated vectors and his/her true z-scores vector including z-scores of items in the gauge set to the server. After finding referrals, the server uses the 1-out-of-n Oblivious Transfer protocol to send them to a that receives only one prediction instead of Y recommendations. Although this method is advantageous for the server in terms of business purposes, it introduces additional computation costs due to the 1-out-of-n Oblivious Transfer protocol. Moreover, it also introduces additional communication costs.

In the second and the third methods, active users also perturb their data as other users do. Note that a only sends the ratings for the items in the gauge set to the server. In the second method (M2), a generates k random numbers drawn from a distribution with μ being 0. a then adds them to his/her ratings in the gauge set, and sends the disguised ratings in the gauge set to the server to get recommendations. In the third method (M3), a creates m_a random numbers, where m_a represents the number of rated items by a. After that, a uses the first k random numbers to disguise his/her ratings in the gauge set. After perturbing ratings in gauge set, a sends them to the server. The last two solutions do not introduce additional communication costs. Additional computation costs are negligible. Due to random numbers, they make accuracy worse compared to the first solution.

4.5 Providing Referrals

After estimating the correlation matrix C' from the masked data, the server clusters the users and finds the lookup tables for each cluster off-line. When an active user a wants a prediction for a single item q or top-N recommendations for his/her unrated items, he/she sends his/her data in a private manner to the server together with the query. The server first projects a's data and finds the representative cluster for a. Then, it finds the estimated value, which we call it P', from the corresponding lookup table. It sends it to a, who can now de-normalize P' and finds prediction p'_{aq} for q, as follows: $p'_{aq} = \overline{v}_a + \sigma_a \times P'$. The server will only be able to know the estimated value of P' because it does not have the mean rating and the standard deviation of a who is looking for prediction. Therefore, it will not be able to learn how much a likes or dislikes q. To provide top-N recommendations, a sends a query stating that he/she is looking top-N recommendations for N_a items, where $N < N_a < m - m_r$, and m_r is the number of items rated by a. The server then computes P' values for all N_a items and sorts them decreasingly. Finally, it selects the first N items and sends the list to user a as top-N recommendations. Since the same mean and the standard deviation are used to de-normalize P' values, the server does not need them to find the sorted list.

5 Overhead Costs and Privacy Analysis

Off-line costs are not critical to overall performance. Therefore, we are interested in additional online costs rather than off-line ones. Since privacy, accuracy, and efficiency are conflicting goals, providing privacy measures makes accuracy and/or efficiency worse. We show how accuracy changes with privacy measures in the subsequent section. We now show how much additional costs introduced due to privacy concerns. The communication cost, in terms of number of communications, does not change due to privacy-preserving schemes when a protects its privacy using the second and the third methods. However, the amount of data to be sent increases due to the appended random values. The communication costs increase due to 1-out-of-n Oblivious Transfer protocol when a uses the first method for privacy-preserving. Although the amount of data to be stored increases due to random values, the server stores the collected perturbed data into the same $n \times m$ matrix. Additional computation costs due to privacy concerns are also small. Moreover, data disguising is performed without the help of a third party. The only additional computation cost is the computations that the server conducts to get rid of the contribution of random values in diagonal entries of correlation coefficient matrix.

The server tries to figure out the actual values of the ratings and the rated items. The server wants to obtain true ratings from masked z-scores. Since users perturb their z-score values, it becomes difficult to obtain true ratings from perturbed z-scores. To obtain such values, the server needs the standard deviations and means of users' ratings, which are only known by the users. It should know the type of the perturbing data, as well. Since each user u employs uniform or Gaussian perturbing data based on θ and r_u, the server can guess that uniform or Gaussian perturbing data is used with probability θ and 1-θ, respectively. The standard deviations of random numbers are also critical. The server does not know such values, because they are uniformly randomly generated over the range $[0, \gamma]$ by the users. It only knows the γ, not σ_u values. The privacy measure should indicate how closely the original value of an item can be estimated from the perturbed data. To quantify privacy introduced the RPT, various measures are employed [1,2]. Privacy introduced due to uniform or Gaussian perturbing data is analyzed in [14] and can be similarly analyzed. As expected, different perturbing data provide various privacy levels. Moreover, with increasing level of perturbation (with increasing σ_u values), privacy improves due to increasing randomness. We can say that our proposed schemes improve privacy because the server does not know the means and the standard deviations of users' true ratings, type of perturbing data, and the standard deviations of random numbers.

We can analyze our proposed method for preventing the server from learning the rated items, as follows: The server does not know the rated items due to appended random numbers. However, it can guess the randomly selected unrated items' cells. We should compute the probability of guessing such cells for the server. The probability of guessing the correct x_{er} value for the server is 1 out of δ. After guessing such value, the server can figure out the number of filled unrated items' cells (d) with the help of empty cells in the perturbed vector because the number of empty cells equals 1-x_{er} percent of unrated items' cells. After finding d, the server can guess the randomly filled unrated items' cells. If the number of filled cells is m_u including rated items' cells and filled unrated items' cells, the probability of guessing d randomly selected unrated items' cells is 1 out of $\mathrm{C}_d^{m_u}$, $\mathrm{C}_d^{m_u}$ represents the number of ways of picking d unordered outcomes from m_u possibilities. Therefore, the probability of guessing the randomly selected unrated items' cells of one user is 1 out of $(\delta \times (\mathrm{C}_d^{m_u}))$.

6 Experiments

To assess the overall performance of our schemes, we performed experiments using real data. We analyzed the outcomes and displayed the final results.

6.1 Data Set and Evaluation Metrics

We performed trials using a well-known real data set, Jester, which is a web-based joke recommendation system, developed at the University of California, Berkeley [11]. In Jester, users rate a core set of jokes, and then receive recommendations about others that they should like. The database has 100 jokes and records of 17,988 users. Almost 50% of all possible ratings are present. The ratings range from -10 to +10, and the scale is continuous. Each user rated the first 10 jokes and these 10 jokes represent the gauge set. The ratings for the remaining 90 jokes are used for generating lookup table. Since Jester is the only available data set, which contains ratings for the gauge set, we performed experiments using Jester only. However, our results based on Jester can be generalized.

Several evaluation criteria for CF have been used in literature [5,10,20]. The most common criteria are the *Mean Absolute Error (MAE)* and the *Normalized Mean Absolute Error (NMAE)*. We also employed them as our choice of evaluation criteria in our testings. The MAE and the NMAE should be minimized. The lower the MAE and the NMAE, the more accurate our schemes are. The MAE is a measure of the deviation of recommendations from their true user-specified values [20]. If $p_1, p_2, \ldots, p_d$ are true ratings, and $p_1^{'}, p_2^{'}, \ldots, p_d^{'}$ are predicted values from disguised data, then $\xi = \{\xi_1, \xi_2, \ldots, \xi_d\} = \{p_1^{'} - p_1, p_2^{'} - p_2, \ldots, p_d^{'} - p_d\}$ represents errors. Therefore, the MAE is computed, as follows:

$$MAE = \frac{\sum_{i=1}^{d} |\xi_i|}{d}. \tag{15}$$

As it is used in Goldberg et al. [10], we also employ the NMAE as an evaluation metric, which can be defined, as follows, where r_{max} and r_{min} represent the maximum and the minimum ratings, respectively:

$$NMAE = \frac{MAE}{|r_{max} - r_{min}|}. \tag{16}$$

6.2 Methodology

We first randomly divided the data set into training and test sets, where training set contains 9,000, while test set includes the remaining 8,988 users' ratings. We used these randomly selected 9,000 users' data as training data. For testing, we randomly selected 5,000 users from test set. For each test user (active user, a), we randomly selected 10 rated items among non-gauge items. We normalized the train and test users' ratings by converting them into z-scores using user-mean votes. We perturbed the private data as explained previously. To obtain trustworthy results, we run data disguising 100 times. Throughout the experiments, we set k at 10 (there are 10 items in gauge set), v at 2, and clustered users into 57 clusters using k-means clustering. For each test user, we withheld one of the test items' rating and tried to find prediction for it. We compared the predictions with true ratings. We computed the MAE and the NMAE values. We finally displayed the final overall average values.

6.3 Experimental Results

Note that we employed user-mean votes rather than item-mean votes to normalize the ratings into z-scores. Then, we used k-means clustering instead of recursive rectangular clustering. Finally, we generated lookup tables based on z-scores. Therefore, we first performed experiments to evaluate the overall performance of the modified and/or simplified Eigentaste algorithm. For this purpose, we used 9,000 users for training while 5,000 test users and randomly selected 10 test items for each test user were used for testing. After finding recommendations for each test user and test item, we compared them with the withheld true ratings. We calculated the MAE and the NMAE values. We displayed the final values together with the results for Eigentaste found by [10] in Table 1.

Table 1. Eigentaste vs. Modified Eigentaste

	MAE	NMAE
Eigentaste	3.740	0.187
Modified Eigentaste	3.334	0.167

As seen from Table 1, our modified Eigentaste gives better results than Eigentaste algorithm. Our proposed variations improves accuracy 10%. In the following, we performed experiments using modified Eigentaste while considering privacy concerns.

After evaluating the overall performance of our proposed alternative variations to the original algorithm, we then performed testings to assess our proposed privacy-preserving Eigentaste schemes. Since we proposed three different schemes to protect a's privacy, we conducted experiments to compare those three schemes in terms of accuracy. Again, we used 9,000 training and 5,000 test users, where we found predictions for 10 test items for each test user. To disguise private data, we set θ at 0.5, γ at 4, and δ at 100. We run data disguising 100 times. We found predictions for withheld items for each a based on perturbed data, where we protected a's privacy using three different methods. We then computed overall MAE and the NMAE values. We displayed overall MAE and NMAE values in Table 2, where M1, M2, and M3 represent the first, second, and third methods, respectively.

Table 2. Comparing Methods for Protecting a's Privacy

	M1	M2	M3
MAE	3.3508	3.4710	3.4807
NMAE	0.1676	0.1735	0.1741

The first method gives the best results, as seen from Table 2. However, since it is based on the 1-out-of-n Oblivious Transfer protocol, which introduce additional communication and computation costs. The second and the third methods provide similar results. Compared to the first method, the additional costs they introduce are negligible. Therefore, we used the second method in the following experiments to protect a's privacy. The MAE, 3.3508, for the first method is very close the one for the modified Eigentaste, which is 3.334. This means that the errors due to model creation based on perturbed data is very small. However, the second and the third methods also give decent results.

To show how accuracy changes with varying numbers of users (n), we performed experiments, where we varied n from 500 to 8,000. We used the same 5,000 test users and found predictions randomly selected 10 test items for each test user. To protect a's privacy, we used the second method. Again, we set θ at 0.5, γ at 4, and δ at 100. We run data disguising 100 times. We calculated the MAE and the NMAE values and display our outcomes in Table 3.

Table 3. Accuracy vs. Varying Numbers of Users (n)

n	500	1,000	2,000	4,000	8,000
MAE	4.678	4.242	3.832	3.624	3.483
NMAE	0.234	0.212	0.192	0.181	0.174

As expected and seen from Table 3, the results are getting better with increasing n values. The reason for this phenomenon can be explained, as follows: In estimation of the matrix C', the relative errors due to random numbers will converge to zero with increasing n values. Although the results are worse for n values less than 2,000, the results for n values bigger than or equal to 2,000 are promising. Therefore, it is possible to provide accurate recommendations without violating users' privacy when there are sufficient data.

To prevent the server from learning the rated items, the users perturb some of their randomly selected unrated items' cells by inserting random values. We hypothesize that accuracy varies with different numbers of perturbed unrated items' cells. Therefore, we conducted experiments using 9,000 and 5,000 training and test users, respectively. We run data disguising 100 times and found recommendations for 10 test items for each test user. We varied δ from 0 to 100. When δ is 0, users do not disguise any empty cells while it is 100, the number of disguised cells is the most. With increasing δ, randomness increases. We computed the MAE and the NMAE values for various δ values and displayed them in Table 4.

Table 4. Accuracy with Varying δ Values

δ	0	35	70	100
MAE	3.4460	3.4567	3.4615	3.4710
NMAE	0.1723	0.1728	0.1730	0.1735

As expected, accuracy slightly becomes better with decreasing δ values due to diminishing randomness. However, the gain is very small. This can be explained with the density of Jester data set. Our scheme gives acceptable results even with larger δ values. Therefore, our privacy-preserving Eigentaste-based schemes provide referrals with decent accuracy with varying δ values.

Accuracy and privacy also depend on some other factors other than n, δ, and the methods to protect a's privacy, such as the perturbing data, level of perturbation, σ selection methods, and so on. Since it has been shown that how accuracy and privacy vary with these factors when the RPT is employed [14], we did not conduct experiments to evaluate our proposed schemes with varying of such factors.

6.4 Summary of Our Results

Our proposed schemes allow the users and the server to adjust the parameters of data perturbation methods to achieve required levels of privacy and accuracy. With increasing γ and δ values, we add more randomness to original data. With increasing randomness, privacy improves, while accuracy diminishes. Therefore, the server and the users should select γ and δ values in such a way to obtain a balance between them. Although our schemes provide recommendations with decent accuracy for γ and δ being 4 and 100, respectively, we can even further improve accuracy by using smaller γ and δ values. On the other hand, privacy diminishes with decreasing such values. Users and the server are able to decide the values of such parameters in such a way to achieve required levels of privacy and accuracy. Although Gaussian perturbing data slightly provides more privacy and accurate results then uniform perturbing data, the results are still very close to each other. The users can select either of them to mask their data while still obtaining good results in terms of accuracy and privacy. Although we propose three methods to protect a's privacy, the second and the third methods should be preferred over the first one. When considering accuracy and efficiency together, the last two methods are better because they do not introduce additional online communication and computation costs while sacrificing little on accuracy. Kargupta et al. [12] state that when standard deviation (σ) of perturbing data is less than or equal to 1, actual data can be predicted from disguised data reasonably well. However, as seen the experiment results, our schemes can provide recommendations with decent accuracy when σ is bigger than 1.

7 Conclusions and Future Work

We showed how to achieve private recommendations using Eigentaste algorithm. We proposed schemes to produce accurate referrals without jeopardizing users' privacy. We evaluated the overall performance of our schemes in terms of accuracy conducting various experiments based on real data. We analyzed the outcomes of the experiments. Moreover, we analyzed the proposed schemes in terms of additional costs and privacy. We will study whether we can employ other clustering methods or not besides k-means clustering. In the future, we will investigate how to improve recommendation qualities by using well-known correlation-based CF algorithms after clustering. We will also study providing private recommendations using Eigentaste algorithm based on partitioned data between various parties.

References

1. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy-preserving data mining algorithms. In: Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Santa Barbara, CA, USA, May 21–23 2001, pp. 247–255 (2001)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD on Management of Data, Dallas, TX, USA, May 15–18 2000, pp. 439–450 (2000)
3. Brassard, G., Crépeau, C., Robert, J.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
4. Canny, J.: Collaborative filtering with privacy. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 45–57 (May 2002)
5. Canny, J.: Collaborative filtering with privacy via factor analysis. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, pp. 238–245 (August 2002)
6. Cranor, L.F.: 'I didn't buy it for myself' privacy and E-commerce personalization. In: Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, Washington, DC, USA, October 30, 2003, pp. 111–117 (2003)

7. Cranor, L.F., Reagle, J., Ackerman, M.S.: Beyond concern: Understanding net users' attitudes about online privacy. Technical report, AT&T Labs-Research, (April 1999), Available from http://www.research.att.com/library/trs/TRs/99/99.4.3/report.htm
8. Ding, C., He, X.: K-means clustering via principal component analysis. In: (ICML'04).Proceedings of the 21th International Conference on Machine Learning, Banff, Alberta, Canada, pp. 225–232 (July 2004)
9. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM 28, 637–647 (1985)
10. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4(2), 133–151 (2001)
11. Gupta, D., Digiovanni, M., Narita, H., Goldberg, K.: Jester 2.0: A new linear-time collaborative filtering algorithm applied to jokes. In: Proceedings of the Workshop on Recommender Systems: Algorithms and Evaluation, 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, CA, USA (August 1999)
12. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: Random data perturbation techniques and privacy-preserving data mining. In: (ICDM'03). Proceedings of the 3rd IEEE International Conference on Data Mining, Melbourne, FL, USA, pp. 19–22 (2003)
13. Kim, D., Yum, B.: Collaborative filtering based on iterative principal component analysis. Expert Systems with Applications 28(4), 823–830 (2005)
14. Polat, H.: Privacy-preserving collaborative filtering. International Journal of Electronic Commerce 9(4), 9–36 (2005)
15. Polat, H., Du., W.: Privacy-preserving collaborative filtering on vertically partitioned data. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 3–7. Springer, Heidelberg (2005)
16. Polat, H., Du, W.: Privacy-preserving top-*N* recommendation on horizontally partitioned data. In: (WI'05).Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence,September 19–22, 2005, Paris, France, (2005)
17. Polat, H., Du, W.: Achieving private recommendations using randomized response techniques. In: Ng, W-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 9–12. Springer, Heidelberg (2006)
18. Polat, H., Du., W.: Effects of inconsistently masked data using RPT on CF with privacy. In: Proceedings of the 22nd ACM Symposium on Applied Computing, Special Track on E-commerce Technologies, Seoul, Korea (March 2007)
19. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.: Application of dimensionality reduction in recommender system–A case study. In: Proceedings of the ACM WebKDD 2000 Web Mining for E-commerce Workshop, August 20, 2000, Boston, MA, USA, (2000)
20. Sarwar, B.M., Konstan, J.A., Borchers, A., Herlocker, J.L., Miller, B.N., Riedl, J.T.: Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, Seattle, WA, USA, November 14–18, 1998, pp. 345–354 (1998)

Secure and Private Incentive-Based Advertisement Dissemination in Mobile Ad Hoc Networks

Alexandre Viejo, Francesc Sebé, and Josep Domingo-Ferrer

Rovira i Virgili University
UNESCO Chair in Data Privacy
Department of Computer Engineering and Mathematics
Av. Països Catalans 26, E-43007 Tarragona, Catalonia, Spain
{alexandre.viejo, francesc.sebe, josep.domingo}@urv.cat

Abstract. Advertisement dissemination is a promising M-commerce application which exploits the capabilities of mobile ad hoc networks to increase the visibility of the products being offered by merchants. The starting point is a merchant who generates an advertisement that is subsequently disseminated by citizens who carry mobile devices acting as network nodes. In this paper we present a novel system where users collaborating in offer dissemination are incentivized with e-coin rewards. Our system is proven to be secure and to preserve the privacy of nodes.

Keywords: Advertisement dissemination, Incentive, M-Commerce, Mobile ad hoc network, Privacy, Security.

1 Introduction

In the recent years, e-commerce has become a valid alternative to classic commerce. This has been possible due to the proliferation of home computers equipped with Internet access, cost reduction, special promotions and the possibility of shopping from our home. In spite of its acceptation, e-commerce has not displaced classic shopping which is still the most popular way of trading [1].

New-generation mobile devices (*e.g.* cell phones, PDAs ...) are enabled with wireless communications technologies which paves the way to a broad range of services based on mobile ad-hoc networks (MANET). These networks are formed by mobile nodes which are connected in a self-organized way without any underlying hierarchical infrastructure. In a MANET, nodes not only send or receive but also route data exchanged between other nodes.

The big proliferation of this kind of mobile devices enables the extension of e-commerce (named m-commerce in mobile networks) to traditional shoppers (*e.g.* the users of physical malls). This blurs the distinction between the online and offline worlds and implies the emergence of new trading models which represent new opportunities and challenges.

In this work we present a new m-commerce application based on advertisement dissemination between the nodes of an ad hoc network established in a

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 185–198, 2007.
© Springer-Verlag Berlin Heidelberg 2007

certain city. In such a system, a merchant generates advertisements related to her products. These advertisements are spread by incentivized volunteers who carry mobile devices and act as network nodes. The high mobility of nodes facilitates the dissemination of advertisements around the city.

Such a system must consider the following aspects:

- Nodes should obtain some reward for disseminating an advertisement. Otherwise, they may refuse to collaborate.
- When incentives for collaborating nodes are established, the system should prevent a malicious node or a malicious collusion of nodes from being able of obtaining higher rewards than due.
- An incentivized system requires accounting the collaboration carried out by nodes. But since a MANET consists of peer nodes operated by users, this accounting information should not allow tracking of users, who should retain their privacy.

In light of the above, an advertisement will contain metadata fields such as the merchant's reward offer per purchase and accounting information corresponding to nodes having collaborated so far in the dissemination of the advertisement. The system ought to be designed so that accounting cannot be maliciously altered but the privacy of the users is guaranteed.

In a MANET, nodes are constantly changing their location. This can cause any pair of nodes to lose direct connectivity any time. In this way, communication systems for MANETs should not depend on centralized authorities that need to be permanently accessible. As to computational capabilities, nodes can be assumed to be able to perform any cryptographic operation.

1.1 Previous Work

There are several proposals in literature which provide incentivized information sharing among mobile devices in a MANET (shared information does not need to be restricted to announcements).

The authors in [5] and [7] propose incentive-based schemes where the network nodes maintain an account with a special node that gives them credit depending on the information they have shared so far. The network nodes can later redeem their credit for money. Nevertheless, both proposals only ensure content integrity. Thus, malicious nodes could claim more credit than they have actually earned.

Providing incentives to intermediate nodes requires a secure way of collecting them. In [4] a lightweight and cheat-resistant micropayment scheme is proposed to stimulate and compensate collaborative peers that devote some of their resources to relay packets for other peers. This scheme focuses on providing a secure and stable channel to exchange data between two peers within an ad hoc network. Intermediate nodes are incentivized to keep this channel operative. However, this work does not address secure dissemination of advertisements across a large set of mobile nodes.

The authors in [6] present AdPASS. This is a system designed to spread digital advertisements among interested mobile nodes in an urban MANET. There are three types of participants in this scheme:

- A *merchant* disseminates digital advertisements within her vicinity. Customer devices learn about advertisements while their customers are browsing around the shop.
- A *customer* device collects advertisements and transmits them to other interested customers while moving around the city. If a customer uses a received advertisement to buy something at the merchant's (the one originating the advertisement), all the users who have co-operated in relaying the advertisement to the buyer receive some bonus points. Such bonus points can later be traded for goods at the merchant's.
- A *mediator* keeps track of the users' accumulated bonus points. The mediator is similar to a central database accessible to both the merchant and the customer. In addition, the mediator acts as an anonymizer to guarantee the anonymity of customers.

AdPASS relies on a *Virtual Bonus Points scheme* to manage the bonus points obtained by a user from dissemination of a certain advertisement. According to this scheme, the merchant initially fixes the total number of bonus points which she is willing to pay for each purchase of the product. Each user who participates in the dissemination decides how many of the remaining bonus points she takes. This decision has an influence on the probability of further dissemination of the advertisement. If a greedy user takes nearly all the bonus points left for a certain advertisement, no other user is likely to be interested in further disseminating the advertisement; this may result in loss of sales opportunities, so a rational user might be expected to take only a fair share of the bonus points allocated per product purchase.

Even though AdPASS is supposed to provide security and privacy to the users who disseminate advertisements, it is not without problems. For one thing, the authors only explain how to get the bonus points but they do not mention how such points are later spent. This issue must be addressed since privacy could be compromised at the time of spending. Besides, this approach requires the users to register with a trusted authority named *mediator* which acts as an *anonymizer* and keeps track of the users' accumulated bonus points. We claim that a good system should not require the presence of a trusted third party (TTP).

On the other hand, the virtual bonus points scheme in AdPASS offers no guarantees of fairness: even though a reasonable behavior can be expected, the fact is that each user disseminating an advertisement can take as many points as she wishes, regardless of how many she actually deserves. Worse yet, collusions are conceivable where colluders exclude other users from dissemination in order to monopolize bonus points. AdPASS must definitely be repaired to thwart those roguish attitudes. Last but not least, the total number of bonus points assigned by the merchant to an advertisement is a *de facto* upper bound on the number of feasible transfers to new disseminators: due to the limited range of MANET nodes, this implies some limitation in the geographical dissemination range and the sales potential.

To summarize, to the best of our knowledge, none of the existing proposals in the literature addresses all the requirements needed to provide advertisement dissemination through MANETs in an effective, secure and privacy-preserving fashion.

1.2 Contribution and Plan of This Paper

In this work, we present a new scheme designed to disseminate the advertisements of a merchant in mobile ad hoc networks. Our system offers incentives to stimulate the collaboration of nodes. Cryptographic techniques are used to prevent manipulation and preserve the privacy of users. Specifically, the AdPASS [6] system is outperformed in the following aspects:

- Security is achieved against (individual or colluding) dishonest nodes trying to modify transmitted advertisements in order to unlawfully increase their share of incentives.
- Privacy is preserved without resorting to any trusted third party. Our system only requires a certification authority (CA) to certify the merchant's public key. In any case, this authority can not disclose users' identities.
- The incentives rewarding a certain purchase are distributed among all cooperating users given on how long they have held the advertisement leading to that purchase before transferring it to another user. This is a fair proposal which does not restrict the advertisement's dissemination range.

This paper is structured as follows. Section 2 presents some background on public key cryptography over Gap Diffie-Hellman groups. Section 3 describes the new scheme. Section 4 contains a security and privacy analysis. Conclusions are summarized in Section 5.

2 Cryptography over Gap Diffie-Hellman Groups

The construction we propose uses multisignatures over a Gap Diffie-Hellman group [2]. Next, we briefly introduce its mathematical background. A Gap Diffie-Hellman (GDH) group G is an algebraic group of prime order q for which no efficient algorithm can compute g^{ab} for random $g^a, g^b \in G$, but such that there exists an efficient algorithm $D(g^a, g^b, h)$ to decide whether $h = g^{ab}$. GDH groups are suitable for public-key cryptography. The secret key is a random value $x \in \mathbb{Z}_q$ and its corresponding public key is $y \leftarrow g^x$. The signature on a message m is computed as $\sigma \leftarrow \mathcal{H}(m)^x$ ($\mathcal{H}$ is a cryptographic one-way hash function). In the rest of the paper we will denote such a signature on m as $\{m\}_x$. The validity of a signature can be tested by checking $D(y, \mathcal{H}(m), \sigma)$.

GDH groups are convenient to compute multisignatures. Given two signatures of the same message m under two different public keys y_1, y_2, a signature of m under the combined public key $y \leftarrow y_1 y_2 = g^{(x_1+x_2)}$ can be obtained as $\mathcal{H}(m)^{x_1}\mathcal{H}(m)^{x_2} = \mathcal{H}(m)^{x_1+x_2}$.

3 Our Proposal

Our protocol assumes the existence of a merchant and several mobile nodes that communicate through a MANET. We assume the existence of dishonest users (who may act individually or in collusion) interested in obtaining a higher reward than that they are entitled to. We do not require the users to be registered with any central entity. Thus, our system is appropriate for very dynamic environments where connectivity to a central entity may not be guaranteed.

Functionally speaking, a user holding an advertisement actively contacts users within her range and sends them the content of the advertisement. Initially, the advertisement is held by the merchant. Some of the contacted nodes may purchase the advertised good and/or be interested in holding the advertisement themselves for further dissemination.

On the occasion of a purchase request, the buyer sends to the merchant the advertisement (if any) which has motivated her purchase; attaching the advertisement entitles the buyer to a discount. The incentives rewarding that purchase are distributed among the nodes in the path from the merchant to the buyer proportionally to the time they have held the advertisement. E-coins are used to pay those incentives.

In order to facilitate the distribution of incentives, when an advertisement is transferred to a new holder, a time stamp indicating the moment of the transfer is added to the advertisement. In this way, when an advertisement comes back to the merchant together with a purchase request, the merchant can ascertain the incentive that corresponds to each collaborating node. The system is totally anonymous, *i.e.*, the information that nodes add to an advertisement does not allow to identify them. Also, different contributions of a node to different advertisements cannot be related. In this way, unlinkability is also provided. Obviously, we are assuming that the appropriate measures are being taken to avoid node tracking by other means (for instance, frequent change of MAC and IP addresses).

The above system is sustainable for the merchant, who never loses money, because incentives are only paid for advertisements which generated a purchase.

In the following subsections, we describe the protocols and procedures that conform our system. They are:

- Advertisement generation
- Advertisement dissemination
- Advertisement transfer
- Advertisement checking
- Advertisement deposit
- Incentive payment

An advertisement dissemination example in given in Subsection 3.7.

3.1 Advertisement Generation

Merchant M has its public key, PK_M, and its digital certificate issued by a Certification Authority, $Cert_{Aut}\{PK_M\}$. We denote by SK_M the secret key corresponding to PK_M.

1. When M wants to promote a product, it generates an advertisement α containing its public key certificate, the offer description and the expiration time of this offer:
$$\alpha = \{Cert_{Aut}\{PK_M\},\ Description,\ ExpirationTime\}$$
This advertisement is signed by M to obtain $\{\alpha\}_{SK_M}$.
2. A node U_i interested in disseminating the advertisement contacts M and receives the following message:
$$\beta = \{\alpha,\ PubKeyChain,\ Multisignature,\ TimeChain\}$$
The fields of β are initialized as follows:
   - $PubKeyChain$ is an ordered list initially left empty;
   - $Multisignature$ is initialized to $\{\alpha\}_{SK_M}$;
   - $TimeChain$ is an ordered list initially containing a single element that is a tuple formed by $Time$ and its signature $\{Time\}_{SK_M}$; $Time$ corresponds to the time this operation has been performed.
3. U_i checks β (see Section 3.4). If all checks are correct, U_i accepts the advertisement from M and starts its dissemination.

3.2 Advertisement Dissemination

Upon accepting an advertisement, U_i informs other nodes about the offer it contains. Due to the inherent mobility in the nodes, U_i is likely to disseminate the offer quite far from M.

Additionally, when U_i contacts a nearby node U_j, U_i asks whether U_j is interested in disseminating the advertisement (our scheme is not linked to any specific framework to perform such initial contact between users, the one presented in AdPASS [6] can be used). If she is, they will start the advertisement transfer. In order to guarantee anonymity and unlinkability, nodes must change their MAC and IP addresses after each contact.

Note that, after an advertisement transfer from U_i to U_j, U_i still holds the advertisement and can continue its dissemination and transfer to other nodes. In this way, the number of nodes disseminating a certain advertisement can grow exponentially.

3.3 Advertisement Transfer

The advertisement transfer protocol requires users U to have a public/private key pair (PK_U/SK_U). To provide unlinkability, this key pair has to be changed after each execution. Before renewing her key pair, a user stores the secret key. This key will be needed in order to receive the incentives (as will be detailed next in Section 3.6).

1. A user U_j interested in an advertisement α held by another user U_i asks U_i to transfer it.
2. U_i appends her public key to the value $PubKeyChain$ in β. This is
$$PubKeyChain' := PubKeyChain \cup PK_{U_i}$$
3. U_i Computes the signature $sig := \{\alpha\}_{SK_{U_i}}$. Then she computes
$$Multisignature' := Multisignature \cdot sig$$
4. U_i obtains the current time, signs it and appends the signed time to the time chain, that is: $TimeChain' := TimeChain \cup \{Time \mid\mid \{Time\}_{SK_{U_i}}\}$ (at the end).
5. U_i generates
$$\beta' := \{\alpha,\ PubKeyChain',\ Multisignature', TimeChain'\}$$
and sends it to U_j.
6. U_i stores the secret key SK_{U_i} and generates a new key pair that will be used at the next transfer.
7. U_j checks β' (see Section 3.4). If all checks are correct, U_j informs other nodes about the offer in β'.

3.4 Advertisement Checking

A user U_i receiving a message β should check its validity prior to accepting it. This is done as follows:

1. Check the validity of $Cert_{Aut}\{PK_M\}$ (obtained from α). This requires checking the signature by the authority, its expiration date and, if possible, its revocation status.
2. Compute the product of all public keys contained in $PubKeyChain$ and PK_M. Let us denote by $GlobalKey$ the result of this operation.
3. Check that $Multisignature$ is a correct signature over α that is validated using $GlobalKey$.
4. Check that $ExpirationTime$ (obtained from α) has not expired.
5. Check that the first element of $TimeChain$ is a correct signature generated by the Merchant.
6. For each key contained in $PubKeyChain$, check that the j-th public key in $PubKeyChain$ can validate the $(j+1)$-th signature in $TimeChain$.
7. Finally, check that the values of elements in $TimeChain$ are sorted in ascending order and that the last element corresponds to the current time.

3.5 Advertisement Deposit

A user U_i interested in the product advertised in β contacts the merchant and buys it. By sending β to the merchant, U_i will obtain the price reduction detailed in β. This price reduction motivates users to deposit advertisements.

3.6 Incentive Payment

Once a merchant sells a product to a customer who has deposited an advertisement, it has to pay the incentives to all users who have collaborated in its dissemination.

The merchant gives a fixed amount of money for each received advertisement. This amount of money is divided between collaborating nodes proportionally to the time each collaborating node has held the advertisement along the path from the merchant to the buyer (see Section 3.8 for details about the model used to reward incentives). This information can be obtained from the values in $TimeChain$. The merchant does not know the identity of the nodes that collaborated in the advertisement distribution. It only knows their public key. For each payment the merchant authorizes her bank to issue an e-coin. Let us assume user U_i (who remains anonymous and is only known by her public key) has to receive an e-coin for a given value v.

The merchant sends a message to her bank indicating that she can issue an e-coin with value v to any person providing password p. Then, the merchant publishes a message in a public repository containing p encrypted with the public key of U_i. This indirect procedure through a public repository is needed because U_i is anonymous and may be temporarily out of range.

Later, U_i checks the repository, decrypts the message and obtains p. Using this password, the bank permits her to obtain an e-coin (through the corresponding e-coin issuing protocol). The e-coin system must be anonymous such as the one proposed by Chaum in [3]. This is because the e-coin may later be spent non-anonymously (for instance, if the purchased product has to be delivered by courier). If the e-coin system was not anonymous, it could be possible to link the identity of the person spending the e-coin to the public key used in the dissemination protocol.

3.7 Example of the Protocol

We next clarify the operation of our dissemination protocol following the communication steps described above. We base our explanation on the graphical example shown in Figure 1.

1. Advertisement generation. The merchant wants to promote a certain product and generates an advertisement and informs about it the users within range. User A is interested in disseminating this advertisement and contacts the merchant to request transfer of the advertisement β. Then A checks the validity of β and starts its dissemination. This occurs at time T_0.
2. Advertisement dissemination. A roams around while informing other nodes she meets about advertisement β. Then, A transfers the advertisement to two interested nodes B and D at times T_0+T_1 and $T_0+T_1+T_2$ respectively. At time $T_0+T_1+T_3$, node B transfers the advertisement to node C.

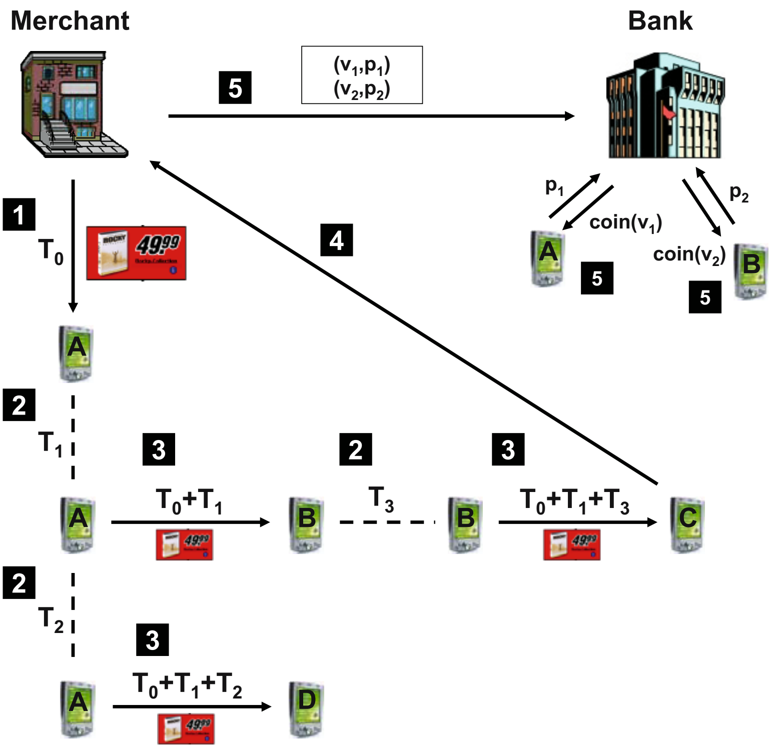


Fig. 1. Graphical example

3. Advertisement transfer. In each transfer, the node which receives the advertisement checks its correctness (see Section 3.4) prior to accepting it.
4. Advertisement deposit. User C is interested in the product advertised in β. Therefore, she contacts the merchant and buys it. By sending β to the merchant, C will benefit from the price reduction detailed in the offer.
5. Incentive payment. The merchant uses the values in the $TimeChain$ embedded in β to determine that A has carried this advertisement during time $T_0 + T_1$ and B has carried it during T_3. Then, the merchant sends a message to its bank indicating that it can issue two e-coins for values $v_1(T_0 + T_1)$ and $v_2(T_3)$ to any person providing passwords p_1 and p_2 respectively. The joint value of those two e-coins is the fixed amount that the merchant is willing to pay for each completed sale of the product. Finally, the merchant publishes p_1 and p_2 encrypted with the public key of A and B respectively in a public repository.

Later, A and B check the repository and obtain their respective password. Then, they contact the bank and obtain their respective e-coin through the corresponding e-coin issuing protocol.

3.8 Comparison to Other Reward Models

As explained before, in our scheme the merchant divides a fixed amount of money between the nodes which have collaborated in an advertisement dissemination. The money earned by a certain node is proportional to the time which such collaborating node has held the advertisement along the path from the merchant to the buyer. We next explain the advantages of this approach in comparison with the model presented in [6] and with a simple model where each node receives money each time it collaborates (this scheme does not consider how long a node has held the advertisement, only if such node has held it or not).

In [6], the merchant fixes an amount of points as reward to a certain advertisement. Each user which collaborates in the dissemination will claim the number of points that she desires. It means that if a greedy user U_i claims too many points, such advertisement will not be disseminated by any other user since it will not have enough remaining points. Thus, it represents a strong restriction in the advertisement's dissemination range. Besides, users are not rewarded in a fair way and this motivates the users to apply strategies of keeping and passing along points instead of collaborating in the dissemination.

The simple model is more fair than [6]. Each U_i which takes part in a dissemination will receive the same amount of money. However it has two main problems:

1. If there is no limit in the number of hops, also there is no limit in the amount of money that the merchant must give as incentives. It represents a major concern for the merchant. We can solve this problem applying a limit but then the advertisement's dissemination range will be restricted like in [6].
2. Since the merchant gives incentives to each user which collaborates, a certain user with n identities can transfer a certain advertisement to herself $n-1$ times (using her $n-1$ alternative identities). At the end of the process, this user will get incentives for each of her n identities.

To solve these two problems we propose to add a second dimension (the time which a user holds an advertisement) to the simple model. Besides, the merchant establishes a fixed amount of money (incentives) that will be divided between the collaborating users. We next explain how our proposal affects the two problems stated:

1. The merchant after each sale divides the money, assigned to pay advertisement dissemination, between collaborating nodes proportionally to the time each collaborating node has held the advertisement. It means that the merchant never loses money. Besides, users will always receive incentives, although a node which has hold a certain advertisement for a little time in comparison with others, will probably get a very small amount of money.
2. A certain user which holds an advertisement within n epoch (interval of time) will get the same amount of money than a dishonest user which has n different identities and holds the advertisement within 1 epoch with each identity.

4 Security and Privacy Analysis

We next explain the adversary model and the possible attacks the system has to be robust against. We refer to such attacks to prove the security properties achieved by our scheme: integrity, authentication and non-repudiation. We also explain how privacy (anonymity and unlinkability) is obtained.

4.1 Adversary Model

In our system, an adversary is any entity or group of entities wishing to disrupt normal system operation or aiming to collect information on nodes who have collaborated in advertisement dissemination. The nodes that can take part in a dishonest coalition are:

- *The merchant.* It may wish to identify and/or trace nodes who collaborate by spreading announcements. It may also repudiate having generated a certain offer.
- *The bank.* It may wish to identify and/or trace nodes who collaborate in message dissemination.
- *Dishonest users.* They may wish to alter advertisements so as to increase the amount of their assigned reward. They may also wish to inject false disrupting data or identify and/or trace other users.

Possible attacks. On the whole, an adversary can try to perform the following attacks:

- Modify the offer description.
- Repudiate having issued a certain advertisement (when the adversary is the merchant).
- Remove the contribution made by some user to message dissemination.
- Issue a fake advertisement.
- Collect incentives corresponding to other users.
- Obtain the identity of a collaborating node and/or profile her by relating different interactions.

4.2 Attacks and Security/Privacy Properties

Modification of an offer description. This attack refers to the integrity property. Offer descriptions are issued by the Merchant, so we assume it does not take part in the coalition. In our system, an advertisement consists of a message with the following structure:

$$\beta = \{\alpha,\ PubKeyChain,\ Multisignature,\ TimeChain\}$$

The advertisement itself is α which contains its public key certificate, the offer description and its expiration time:

$$\alpha = \{Cert_{Aut}\{PK_M\},\ Description,\ ExpirationTime\}$$

Integrity of the offer description is ensured since α is signed by the merchant (this signature is included in the *Multisignature* field) and the signature scheme is unforgeable.

Advertisement repudiation. In our scheme, the merchant cannot repudiate having issued an advertisement since it has been signed and the signature on it is verifiable with a certified public key.

Note that, since collaboration in advertisement dissemination is anonymous, users do not need to repudiate having collaborated.

Removal of user contribution to dissemination. Another integrity aspect to be considered is whether users having contributed to the distribution of an advertisement can be unlawfully dropped and forgotten about. Let us assume an advertisement coming from merchant M that has been distributed by users $U_1, U_2, \ldots, U_n$. Let us assume that an intruder wishes to remove U_i from β. The intruder must remove the public key PK_{U_i} from *PubKeyChain* and remove $\{Time \mid\mid \{Time\}_{SK_{U_i}}\}$ from *TimeChain*. Both removals can be done without any difficulty.

The difficulty for the intruder is to alter the *Multisignature* field. This field contains the value

$$Multisignature = \mathcal{H}(\alpha)^{SK_M + SK_{U_1} + SK_{U_2} + \ldots + SK_{U_n}}.$$

The intruder must be able to obtain

$$Multisignature' = Multisignature \cdot (\mathcal{H}(\alpha)^{SK_{U_i}})^{-1}$$

Since discrete logarithms are hard to compute in a GDH group, the only way to obtain such value by an intruder is to get the *Multisignature* field before U_i's contribution. This value can only be obtained if the intruder contacts directly the user who transferred β to U_i. This cannot be done due to the anonymity of the system.

Issuance of a fake advertisement. This attack refers to the authentication property. Our system requires the merchant to sign advertisements using a public key certified by an accepted authority. Generation of a certain advertisement that will be accepted as authentic coming from a valid merchant M, requires knowledge of its private key SK_M. As long as this secret key is not compromised and the signature scheme is unforgeable (a valid signature can only be computed if the secret is known) the system provides authentication and remains secure against this attack.

Collecting incentives from other users. This situation refers to the authentication property too. In our system, E-coins given as incentives can only be collected by the users who have earned them. This is ensured by the incentive payment procedure. During this procedure, the merchant publishes the password required to obtain an e-coin encrypted with a public key whose corresponding private key is only known by the authentic user. In this way, only the authentic user will be able to obtain this password and request the e-coin.

Disclosure of the identity and/or tracing of users. This attack compromises the privacy of the users. This property consists of two components that must be guaranteed:

- *Anonymity*: Interaction with the system should not reveal the identity of the user.
- *Unlinkability*: It should not be possible to relate different interactions by the same user.

The anonymity of users collaborating in the dissemination of an advertisement is ensured because they simply are requested to provide a public key that does not reveal anything about their identity. Obtaining the password that permits to request an e-coin does not require the user to identify herself either. Finally, an anonymous e-coin system like [3] also provides anonymity when obtaining and spending an e-coin.

Unlinkability is provided if users use a different key pair each time they perform an advertisement transfer. Each user U is able to randomly generate a new public/private key pair (SK_U/PK_U) at will and there is no connection between all the key pairs used by a certain user. Thus, two different public keys from the same user cannot be related by an observer.

5 Conclusions

We have presented a new scheme designed to disseminate advertisements through mobile ad hoc networks. Our scheme outperforms the current proposals in literature by offering security and privacy without requiring the participation of any trusted third party (except for a certification authority that certifies the merchant's public key). In addition to that, we propose a new approach to reward nodes that collaborate in the dissemination according to the time they have been holding an advertisement. Such proposal does not bound the number of transfers for an advertisement (and thus its spreading range) and rewards collaborative nodes with e-coins proportionally to their task.

Disclaimer and Acknowledgments

The authors are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. This work was partly supported by the Spanish Ministry of Education through projects SEG2004-04352-C04-01 "PROPRIETAS" and CONSOLIDER CSD2007-00004 "ARES", and by the Government of Catalonia under grant 2005 SGR 00446.

References

1. Euromonitor International. Global Market Information Database (GMID), http://www.gmid.euromonitor.com
2. Boldyreva, A.: Efficient threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
3. Chaum, D.: Privacy Protected Payments: Unconditional Payer and/or Payee Anonymity. In: Smart Card 2000, pp. 69–92 (1989)
4. Pan, J., Cai, L., Shen, X., Mark, J.W.: Identity-based secure collaboration in wireless ad hoc networks. Computer Networks 51(3), 853–865 (2007)
5. Rajasekaran, H.: An incentive based distribution system for DRM protected content using peer-to-peer networks. In: 1st International Conference on Automated Production of Cross Media Content for Multi-channel Distribution, pp. 150–156 (2005)
6. Straub, T., Heinemann, A.: An Anonymous Bonus Point System For Mobile Commerce Based On Word-Of-Mouth Recommendation. In: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 766–773 (2004)
7. Vishnumurthy, V., Chandrakumar, S., Sirer, E.: Karma: A secure economic framework for p2p resource sharing. In: Workshop on Economics of Peer-to-Peer Systems (2003)

Verifiable Internet Voting Solving Secure Platform Problem[*]

Mirosław Kutyłowski and Filip Zagórski

Institute of Mathematics and Computer Science
Wrocław University of Technology

Abstract. We present a voter verifiable Internet voting scheme which provides anonymity and eliminates the danger of vote selling even if the computer used by the voter cannot be fully trusted. The ballots cast remain anonymous - even the machine does not know the choice of the voter. It makes no sense to buy votes - the voter can cheat the buyer even if his machine cooperates with the buyer. Nevertheless, the voter can verity that his vote has been counted.

Keywords: electronic voting, vote selling, coercion resistance, anonymity.

1 Introduction

Recently, there is a lot of public interest in electronic voting schemes. There are expectations that in a near future modern technologies may significantly improve current election procedures. However, while it became evident that traditional procedures have many inevitable flaws, it is still an unsolved problem how to design electronic voting schemes that fulfill all security demands – not only on the level of a cryptographic protocol, but also concerning the voting equipment as a potential adversary. In this paper we concern the problem of casting a vote via Internet, which is the most challenging problem.

Problems and Risks of the Internet Voting. To some extent anonymity can be achieved by traditional voting on paper ballots. (Of course, there is no longer guarantee that the ballots do not contain hidden features invisible for the voter.) Electronic ballots are much harder to handle: if the ballots are identical, then there will be plenty of ways to attack the system by casting additional votes. If the ballots are unique, then they might be used for uncovering voters' preferences and for vote selling. If they contain random values, then these values may be used to leak secrets. Another practical issue for Internet voting is voter authentication, but we assume that the voters can authenticate themselves with digital signatures.

Verifiability of the election results is one of the major issues for electronic voting: while for the paper ballots there are some procedures against election frauds

[*] supported by Polish Ministry of Science and Higher Education, project 3 T11C 011 26.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 199–213, 2007.
© Springer-Verlag Berlin Heidelberg 2007

(they work as long as the commissions are honest), electronic voting is virtual and the voter may distrust the security mechanism of mixing and counting the votes. Therefore, one of the important features would be to provide the voter a (printed) trace that enables her to check that her vote has been counted and included in the final result. This concept of voting receipts is a central feature in many schemes (see for instance [1]). However, it is also a major problem for system design. At the same time two requirements should be satisfied:

- a receipt must convince a voter that her ballot was properly counted,
- a receipt must not reveal voter's choice.

Vote selling is the most important problem for Internet voting with profound consequences. Unlike in the case of the traditional voting process, buying votes might be very efficient, non-risky and performed with no direct supervision of the buyer. Simply, the voter downloads and installs a special program that supervises his voting activities on his computer. This software sends appropriate information in an encrypted form to some unknown remote server. Finally, the voter receives some reward. One may try to guard voter's PC against such programs, but this seems to be hopeless. Even if it would be possible for today's operating systems (it is not), an overwhelming majority of the users will not change the operating system or make efforts to reconfigure it only for the sake of Internet voting. Moreover, a voter may want to sell a vote. In this case he would not install security system or he would unmount them, if they were already deployed. Necessary tools would be provided by vote buyers.

Systems in which vote buying is easy are extremely dangerous due to the cost of potential vote-buying systems compared to the cost of election campaign.

(Un)trusted Platforms. Voting software and hardware are practically *black boxes*. Even if they have been checked, the voters has no real guarantee that the systems checked are identical with the systems installed on the voters' PC. Even if it is so, the voter may distrust the authorities involved in the audits. In particular, one may fear that the authorities could have installed some features for supervising a voter or manipulating his or her vote.

Previous Solutions. Many electronic voting systems have been proposed so far. One can classify them into three groups:

- **Paper Based** - a voter votes in a booth on a paper ballot which is scanned and then electronically counted by the system [2,4,16],
- **Machine Based** - in a booth a voter operates some voting machine [1,12,14,15],
- **Internet Based** - a voter uses her PC to cast a vote [9,13].

Many schemes are receipt-free [9,13], but assume that that the machines used for voting are honest. This approach seems to be unsuited for Internet voting – one would require a detailed audit at least of the operating system and of the application used for voting. Such a verification of voter's hardware and software is practically infeasible.

In [8] a model of electronic voting schemes have been introduced. In the same paper, a coercion-resistant voting system was presented. Still, presented scheme does not take into account that cryptographic operations cannot be performed by a a voter herself ("Voter V_i includes non-interactive proofs of knowledge of $\sigma_i \cdots$"), so in fact, indirectly assumes honesty of the hardware and software used by the voter.

The problem of untrusted voting machines can be partially solved with receipts. Chaum [1] presented a solution for which a voter gets a receipt proving that her vote was counted and at the same time it is meaningless for anybody else. Hence the voter cannot convince a buyer about the vote cast. Later other schemes with receipts were proposed. All they use a two stage verification. First, a voter can check that her vote appears on a certain bulletin board. The second stage should convince her that her vote was properly processed by an array of mix-servers. Two major techniques are used here: Randomized Partial Checking [10] and Neff's zero knowledge proof procedures [15].

Many Internet voting schemes allow a voter to cast a vote only once (or from a single machine). This makes vote selling very easy: a machine may have a special software installed that monitors voting activities and reports them to the buyer. A solution of this problem was implemented in Estonia: a voter can revoke her electronic ballot and cast a traditional paper ballot. Each electronic ballot is signed digitally by the voter, so it is possible to check which ballot has to be removed (the signatures are removed before decryption of the ballots starts). The main problem of this system is that it provides no verifiability and that vote buying still makes sense (only a fraction of sellers will revoke the vote).

Klonowski *et al.* [12] proposed a quite different scheme for voting machines. Each vote contains two parts, each part consists of two halves. One part contains an encoded vote, the other part contains a random identifier. The halves of each part should appear after the final decoding, lack of any half is an evidence of a fraud during mixing and decoding. Each of the halves is processed separately and the processing servers cannot link them together until the final decoding.

In case of [1, 12] the voting machines must be trusted to a certain degree - they know the preferences of the voters; still, they cannot change them. No solution that would work regardless of dishonest voter's PC has been designed so far. Current research concentrates rather on providing a secure environment (like Trusted Platform proposals) - however this approach has basic limitations.

Internet Voting - Threats: Let App denote the voting application run on the PC of a voter. For Internet voting protocols, there are three main types of threats:

Type A: A voter can be cheated by App:
(1) App can cast a vote for a different *specified* candidate,
(2) App can cast a vote for a random candidate,
(3) App can cast an invalid vote.

Type B: A voter may want to sell a vote:
(1) a vote buyer/coercer is physically present when a voter is casting a vote

via Internet or he can impersonate the voter,

(2) the buyer/coercer provides appropriate application to be run on a machine used for vote casting (to monitor a voter's choice).

Type C: A voter can be cheated by a voting system that implements gathering the ballots, decoding them and computing the results. This problem concerns the protocols which do not provide verifiability:

(1) lack of global verifiability (schemes without audit procedures),
(2) lack of local verifiability (schemes which does not allow a voter to check if her vote was counted as intended).

From a practical point of view the threats of type A and type C are much different. In case of a system concerned in point C detailed audit procedures may take place. On the other hand, there is no control over voter's PC used for Internet voting. In particular, the voter can use any software as App that yields an output according to the specification of the protocol.

Our Results: We design a protocol that generalizes the scheme from [12]. Let us list the main technical features of our proposal:

1. Each encoded ballot is processed through a sequence of tallying authorities that perform mixing and partial decoding; if at least one of these tallying authorities is honest, then the vote remains anonymous.
2. While casting a vote the user obtains a digital receipt that can be used to check if his vote has been properly processed. If a single ballot has been manipulated, then it become detected with a fairly high probability and at least one of the cheating authorities can be identified.
3 The receipt and the transcript of the voting session on the computer of the voter do not suffice to determine the preferences of the voter. Before casting a ballot the voter obtains a short message through an independent communication channel that is hidden for the machine used for voting.
4 A voter can change his decision by casting another ballot, which cancels the previous vote. Both ballots: the first one and the canceling one appear in the final tally, but they cannot be linked together.

It follows that we combine two properties that are somewhat contradictory: a voter can be convinced that his ballot has been counted, but simultaneously buying votes does not make sense. Even if the buyer supervises the computer of the voter (and can see what the voter is doing on this PC), he cannot be sure that the vote will not be revoked later from another machine.

2 Background of Encoding Techniques

RSA-RE Ciphertexts and Signatures. Let us recall a construction of ciphertexts that may be signed and re-encrypted later together with the signature.

The idea has been already used for e-voting [12], and originates from [6,11]. The main advantage of re-encryption is that it allows certain instant verification of the mixing process without revealing anything about the plaintexts.

Key Setup and Ciphertext Creation. Let $N = pq$ be an RSA number, and let g be an arbitrary generator of a cyclic subgroup $G \subseteq \mathbb{Z}_N^*$ such that discrete logarithm problem is hard in G. We skip the notation "mod$N$" whenever operations within $\mathbb{Z}_N$ are concerned.

The authority responsible for vote creation chooses e, which is co-prime with $\varphi(N)$ and d such that $e \cdot d = 1 \bmod \operatorname{lcm}(p-1, q-1)$. Then d is the private signing key, whereas e is the public key for signature verification. The key $\hat{g} = g^d$ is published.

Assume that each ballot has to be processed by λ mix servers before getting decrypted. For $1 \leq j \leq \lambda$, let y_j be the public key (for encryption) of the jth mix, and let x_j be the corresponding private key, where $y_j = g^{x_j}$. Every server obtains also a public key for signature verification, which is equal to $\hat{y}_i = y_i^d$.

A ciphertext of m is created with a randomly chosen k_1 and has the following form:

$$(\alpha, \beta, \gamma, \delta) := (m \cdot (y_1 \cdot \ldots \cdot y_\lambda)^{k_1}, g^{k_1}, m^d \cdot (\hat{y}_1 \cdot \ldots \cdot \hat{y}_\lambda)^{k_1}, \hat{g}^{k_1}) \ . \tag{1}$$

We call it an RSA-RE-onion, since there are many "layers" of encryption and we have to remove these layers in order to decode it.

Decoding Process. When after some decoding and re-encryption such a ciphertext is delivered to mix i, it has the following form:

$$(\alpha_i, \beta_i, \gamma_i, \delta_i) = (m \cdot (y_i \cdot \ldots \cdot y_\lambda)^{k_i}, g^{k_i}, m^d \cdot (\hat{y}_i \cdot \ldots \cdot \hat{y}_\lambda)^{k_i}, \hat{g}^{k_i}) \ .$$

The following operations are executed with r_i chosen at random:

$$\begin{aligned} &(\alpha_{i+1}, \beta_{i+1}, \gamma_{i+1}, \delta_{i+1}) := \\ &\quad (\alpha_i / \beta_i^{x_i} \cdot (y_{i+1} \cdot \ldots \cdot y_\lambda)^{r_i}, \beta_i \cdot g^{r_i}, \gamma_i / \delta_i^{x_i} \cdot (\hat{y}_{i+1} \cdot \ldots \cdot \hat{y}_\lambda)^{r_i}, \delta_i \cdot \hat{g}^{r_i}) \ . \end{aligned}$$

It is easy to see that we get the following tuple with $k_{i+1} = k_i + r_i$:

$$(m \cdot (y_{i+1} \cdot \ldots \cdot y_\lambda)^{k_{i+1}}, g^{k_{i+1}}, m^d \cdot (\hat{y}_{i+1} \cdot \ldots \cdot \hat{y}_\lambda)^{k_{i+1}}, \hat{g}^{k_{i+1}}) \ .$$

Observe that for re-encryption we need to know only the public keys concerned.

Signature Verification. If a RSA-RE-onion signature is correct, then for some k we have $\alpha = m \cdot y^k$, $\gamma = m^d \cdot \hat{y}^k$, so $\gamma = \alpha^d$. Similarly, $\delta = \beta^d$. So we say that the outcome of verification is positive iff $\alpha = \gamma^e$ and $\beta = \delta^e$.

Notation. Obviously, the first two parts of an onion, (α, β), is a regular ElGamal ciphertext obtained for the public key $y_1 \cdot \ldots \cdot y_\lambda$. We will write $ue(m)$ for a RSA-RE-onion of a message m, and $e(m)$ for its first two components corresponding to an ElGamal ciphertext.

Raising to a Power. Let us observe that one can raise m hidden in $ue(m)$ to an arbitrary power l without destroying the signature. Indeed:

$$ue(m)^l = (m^l \cdot (y_1 \cdot \ldots \cdot y_\lambda)^{k \cdot l}, g^{k \cdot l}, m^{d \cdot l} \cdot (\hat{y}_1 \cdot \ldots \cdot \hat{y}_\lambda)^{k \cdot l}, \hat{g}^{k \cdot l}) \ .$$

The last expression is $ue(m^l)$, an RSA-RE-onion of a message m^l, with the exponent $k \cdot l$ used for encryption.

Opening an Onion. It can be checked that an onion $(\alpha, \beta, \gamma, \delta)$ has been created according to formula (1) by *opening* it. Namely, the party that has created the onion reveals the exponent k_1 used. Note that no private keys are necessary.

Coupled Ciphertexts. Assume that two ciphertexts for two different recipients have to be processed together, but in the meantime they have to be re-encrypted. If we do it in a standard way, then the link between the ciphertexts considered is lost after re-encryption and one could replace one of these ciphertexts. However, we can *couple* the ciphertexts of m_1 and m_2 by constructing: $(m_1 \cdot \alpha^k, m_2 \cdot \beta^k, g^k)$, where k is chosen at random, and α, β are the public keys of the addressees of the ciphertexts. We can re-encrypt such a ciphertext by multiplying its components by, respectively, α^h, β^h, g^h for a random h. It is easy to see that use of the same k for encrypting m_1 and m_2 does not introduce any security risks, since a procedure to break the scheme (even with a private key corresponding to β) would yield a method to break ElGamal ciphertexts created with α.

3 Scheme Construction

Since the number of details in the final scheme might be confusing, we present a family of schemes, each time discussing improvements introduced. This should explain the final construction; in the next section, one can find a (high level) step-by-step description of the protocol.

Model. The physical setup of the system is following. Voters use their PC's which are connected to Internet. PC's are running a voting application - App. PC's are also equipped with smart card readers for implementing trustworthy digital signatures. A specialized server named BGS (Ballot Generation Server) is responsible for preparing ballots. A registration server named RS is responsible for verifying signatures of the voters. There are public bulletin boards of authorities responsible for mixing and decoding votes.

Version 0 - Estonian like Solution. Let $y_1, \ldots, y_\lambda$ denote the public keys of the authorities responsible for mixing and decoding the ballots. App prepares a ballot containing m as an ElGamal ciphertext

$$(m \cdot (y_1 \cdot \ldots \cdot y_\lambda)^k, g^k)$$

for a random k. Such an encrypted ballot, signed digitally by the voter, is sent to the bulletin board. At the end of an election day: each signature is verified and it is checked that no voter cast more than one vote. Then decoding and mixing the ballots is performed by the tallying authorities after stripping off the signatures. Correctness of decoding should be checked, e.g. with RPC ([10]). An electronic ballot can be revoked by the voter in a polling station before the end of the election day.

Advantages: *(A.0.1)* If at least one tallying authority is honest, then voter's preferences remain hidden.

(A.0.2). A buyer cannot be sure that the voter will not revoke a vote after selling it (however, the buyer may pay after checking that the buyer has not revoked his e-ballot).

Disadvantages: *(D.0.1)* All threats of type A apply. A fraud cannot be detected, since the random exponent k should be erased immediately in order to protect voter's privacy.

(D.0.2). Practically, the threats of type B apply - revoking personally is tedious and indicates that the voter might be a (dishonest) vote seller.
(D.0.3). Local verification is impossible (threat C.2), only global verifiability can be implemented.

Version 1 - Locally Verifiable Scheme
We apply an idea from [12]. A ballot consists of four ciphertexts: two of them encode the voter's choice (just the identifier of the candidate, the same for all ballots), the next two encode an identifier known to a voter and used for verification. The order of the ciphertexts in the ballot is random. After the first decoding and re-encryption all ciphertexts are permuted at random so that the link between the ciphertexts from the same ballot is lost.

Advantages: All advantages of Version 0 are preserved. Additionally, the scheme is immune against threats of type C:

(A.1.1). Ballot decoding process is in some extent locally verifiable - a voter can check that the identifier of his ballot appears on the final list.
(A.1.2). Removing, modifying and adding one vote can be detected with probability at least $\frac{5}{6}$ in the case of the first tallying authority – equal to the probability of removing 2 parts encoding a vote without removing 2 parts encoding an identifier ($= 1 - \frac{2}{4} \cdot \frac{1}{3}$). The probability that m manipulated ballots remain undetected is lower than $(\frac{1}{4W})^m$ in case of the remaining authorities, where W is the number of votes cast [12].

Disadvantages: The scheme is not immune against threats of type A and B.

Version 2 - Securing the Scheme Against a Dishonest App
Let us assume that there is N candidates. The ballots are created by an independent authority, so called *Ballot Generating Server (BGS)*; the following steps are executed in order to create a single ballot:

(1) BGS generates N 4-tuples of onions, where each tuple is generated as for Version 1 of the protocol and the ith tuple encodes a vote for the ith candidate.
(2) The list of 4-tuples is shifted circularly by a random shift s.
(3) BGS sends the tuples to the PC of a voter.
(4) BGS sends the shift s through an independent channel directly to the voter.
(5) The voter indicates which of the 4-tuples should be cast as the vote.
(6) App re-encrypts the tuple chosen, permutes at random the onions obtained and presents them to the voter for signing.
(7) The voter signs the onions and App sends them to the bulletin board.

Advantages: The scheme is still immune against threats of type C. Additionally:

(A.2.1). Voter's PC does not know voter's preferences (as long as PC and BGS do not collude) – it is the first step to eliminate threats of type A.1.

Disadvantages: *(D.2.1)* PC can change the vote at random - it can send for signing a different tuple than indicated by the voter. Nothing will be detected. However, App cannot change the vote to a vote for a chosen candidate.

(D.2.2). The voter has to trust BGS - the threats of type C apply in case of BGS.

(D.2.3). App may pretend that it is taking onions from BGS, but in fact generate them itself. So all threats of type A still apply.

Version 3 - Enforcing Cooperation with BGS

Instead of onions, BGS creates RSA-RE-onions. Two ciphertexts from a 4-tuple indicating a vote for a candidate i encode the plaintexts c_i and c'_i, where c_i, c'_i are chosen at random from G for all ballots (so in particular, the discrete logarithms of c_i and c'_i with respect to g are unknown).

Advantages: *(A.3.1).* At any level of decoding it can be checked that an onion originates from BGS (by verifying signature that will be inserted into an onion). Hence App cannot create own onions. Together with the property (A.2.1) it prevents threats of type A.1.

(A.3.2). The RSA-RE onions do not eliminate manipulations through raising to power. However, it is unknown how to make a valid vote for another candidate from a vote for candidate i. Together with property (A.2.1) it guarantees that App cannot create random valid votes. So threats of type A.2 are prevented.

Disadvantages: The scheme is immune against threats of type B.2, but threats of type B.1 still apply. Threats of type C exist with respect to BGS.

Version 4 - Securing the Scheme Against Malicious BGS

In order to enforce honest behavior of the BGS, the following additional steps are executed:

(1) Instead of one *card* containing N 4-tuples, two cards are presented on request by the BGS to App (optionally more than two). The onions are still not composed as RSA-RE-onions (there is no signature inside). However, each card is signed digitally in a conventional way for non-repudiation purposes.

(2) Together with the cards App receives cryptographic commitments of:

(a) the shifts used for both cards,
(b) the identifiers.

The commitments can be printed by the voter or mailed in a traditional way on scratch cards.

(3) The shifts are sent to the voter through a secure channel (without involvement of her PC).

(4) The voter chooses one of the cards for checking.

(5) BGS responds with values opening the commitments and onions for the card

requested. Moreover, BGS creates RSA-RE-onions from the onions of the second card.
(6) App presents the result of opening to the voter, who checks the results.

Advantages: As Version 3, it is immune against threats of types A.1, A.2, B.2. Additionally:

(A.4.1). Even a single fraud of BGS becomes detected with a constant probability. So the scheme is immune against threats of type C as long as BGS and App do not cooperate.

Disadvantages: Still, the scheme does not prevent threats of type B.1 and A.3. Unfortunately, this solution enables effective vote selling (B.2): A vote's seller can input obtained shifts to App, then App (not the voter!) chooses a card for voting and verifies the shifts obtained from the voter once BGS reveals the shifts from the unused cards.

Version 4a - Securing against vote seller. BGS, generates s_1, s_2, and derives shift, as $s = s_1 + s_2$. Instead of sending commitment of s to an App, BGS sends commitments of s_1 and s_2. At the same time, BGS sends values of s, s_1, s_2 of each card to a voter. For each card only one of the commitments of values s_1, s_2 is revealed to App. The choice is made deterministically, based on a deterministic signature of a voter under a publicly known value z.

Advantages: Now, a voter cannot prove to App her shifts, because she can check in advance which shift-components will be revealed by BGS. Still, BGS cannot cheat, because it does not know the voter's signature under z. It eliminates threat $B.2$.

Version 5 - Ballot Revoking
Instead of a set of N 4-tuples (each tuple for a different candidate) we have two 4-tuples for each candidate: a *voting tuple* and a *revoking tuple*. The voting tuple for candidate i contains ciphertexts of c_i and c'_i, while two ciphertexts of the revoking tuple for candidate i encode d_i and d'_i, where d_i, d'_i are chosen similarly as c_i and c'_i and serve as "anti-votes" canceling c_i, c'_i. The identifiers in the revoking tuple are chosen independently of the identifiers contained in the voting tuple.

For the purpose of revoking the following additional steps are executed:

(1) *Registration Server* (RS) sends a random challenge r to App,
(2) App derives a symmetric key K from the voter's signature of r (a deterministic scheme like RSA has to be used),
(3) App re-encrypts the revoking tuple and encrypts it with K, the resulting ciphertext is signed by the voter,
(4) App sends the ciphertext of the revoking tuple together with the voter's signature to RS.

At a due time, when the voter decides to revoke his ballot, the following steps are executed (the voter may use a different PC):

(1) the voter signs r once more and sends the signature to RS,
(2) the RS checks the signature, derives K, retrieves the revoking tuple and puts it on the bulletin board just like a voting tuple.

Advantages: *(A.5.1)* Even if a buyer is present when a vote is cast, he cannot be sure if a voter does not revoke this vote and casts another vote later (from a different PC). So the threat type B.1 is minimized.

Disadvantages: Still, the scheme does not prevent threats of type A.3. Moreover, App may cheat and send to the bulletin board a different revoking tuple than it should. In particular, a voter may create own App in order to revoke votes for a candidate, she hates.

Version 6 - Enforcing App to Play Fair
Version 5 of the algorithm requires a pair of 4-tuples per candidate in a ballot obtained from BGS. We change it: now we have two pairs of 4-tuples with identical contents for each candidate - let us call them the *lower* and the *upper* pair. To each pair, BGS attaches two encrypted *marks*, the first mark is either A or $\overline{A}$, the second *hidden* mark is either X or $\overline{X}$. For each candidate the BGS assigns mark A to the upper pair and mark $\overline{A}$ to the lower pair or vice verse, each option with probability $\frac{1}{2}$, independently of the choices for other candidates. Similarly, BGS assigns hidden marks X and $\overline{X}$ to the pairs, independently from other choices.

Now let us describe how to attach the marks chosen for each pair. We use two public keys of the bulletin board, say γ and χ. Then to each onion from a pair of 4-tuples we attach the marks chosen. Namely, instead of an ElGamal ciphertext $(m \cdot (y_1 \cdot \cdot y_\lambda)^k, g^k)$, BGS creates

$$(m \cdot (y_1 \cdot \ldots \cdot y_\lambda)^k, M_1 \cdot \gamma^k, M_2 \cdot \chi^k, g^k)$$

where M_1 and M_2 are the marks chosen for this pair. (In fact, the RSA-RE signatures have to be attached to the resulting tuple.) As already seen, such a coupled ciphertext can be re-encrypted.

The following additional steps are executed while casting a vote:

(1) Together with commitments the voter obtains information about configuration of A and $\overline{A}$ marks in the ballot (and no information about configuration of X and $\overline{X}$ marks).
(2) After choosing a candidate the voter indicates which pair of marked 4-tuples to use: the upper or the lower pair corresponding to her candidate. The ciphertexts chosen are re-encrypted, signed traditionally and sent to RS. The marked revoking tuple is encrypted with the key K, and sent to RS as before.
(3) RS decrypts the first mark (either A or $\overline{A}$) and attaches it to the voting tuple published on the first bulletin board.
(4) The voter checks whether the right mark is published together with her voting tuple.

When a voter demands to revoke her vote, the following additional steps are executed:

(1) the bulletin board decrypts the second mark from the voting tuple obtained from the voter,
(2) after decrypting the ciphertext containing the revoking tuple the bulletin board retrieves both marks attached. The first mark is published together with the revoking tuple.
(3) The bulletin board checks whether both marks are the same for the voting tuple and for the revoking tuple. If not, then a fraud of App is detected.

Advantages: *(A.6.1)* Since there is a voting tuple with mark A (respectively, $\overline{A}$) for each candidate, marks published on the bulletin board do not betray the choices of the voters even to BGS.

(A.6.2). If App performs the steps of the protocol for a different candidate than indicated by the voter (as described by threat A.2), then it has to choose either the upper or the lower pair of this candidate. With probability $\frac{1}{2}$ the chosen pair is wrong- it has a mark different than expected by the voter. So App cannot send a random vote instead of the one chosen by the voter without a fair chance of being detected.

(A.6.3). The voter may collude with App and indicate a revoking tuple for a different candidate than chosen for a voting tuple. The voter can choose a tuple with the correct first mark. However, the voter does not know the hidden mark, so with probability $\frac{1}{2}$ the fraud attempt will be discovered while revoking the vote. Of course, we can increase the chances to detect a fraud: instead of a lower and an upper pair, we might have 2^k pairs for each candidate, where each k-bit mark is assigned to exactly one pair. This decreases the chances of an unnoticed fraud to 2^{-k}.

4 Scheme Construction - Details

Building Blocks
During system setup we choose $c_0, c_1, \ldots, c_{N-1}, c'_0, c'_1, \ldots, c'_{N-1}$ – the values corresponding to the voting options (e.g. list of candidates). Similarly, c_N, $c_{N+1}, \ldots, c_{2N}, c'_N, c'_{N+1}, \ldots, c'_{2N}$ is the list of the values corresponding to the anti-votes, namely anti-vote c_{N+m} or c'_{N+m} corresponds to the vote for candidate/option m. One of the values corresponds to an invalid vote, i.e. $c_0 = \mathit{void}$ $= c'_0$, to allow voters cast invalid votes.

Card. For every card, random s_1, s_2 define a random cyclic shift: $\pi(j) = j + s \bmod N$, for $s = s_1 + s_2$. Two pairs of random identifiers are used: (id, id') and (rid, rid'). The identifiers are coupled in some way (for instance $id' =$ $\mathrm{sign}_{BGS}(id)$, $rid' = \mathrm{sign}_{BGS}(rid)$).

The jth row of a card contains an upper and a lower part. Each of them contains six ciphertexts for casting a vote and six ciphertexts for revoking a vote. Each 6-tuple contains 2 ciphertexts of marks: either of A or of $\overline{A}$ and either of X or of $\overline{X}$. The choice of marks is indicated by random N-bit sequences T_A and T_X. For $t \in \{A, X\}$, $f_{u,t,j} = t$, if $T_t(j) = 1$, and $f_{u,t,j} = \overline{t}$, otherwise. For the lower part, $f_{l,t,j} \in \{t, \overline{t}\}$ and is different from $f_{u,t,j}$.

For $x \in \{u, l\}$ (u, l stand for *upper* and *lower*), row i contains the following 6-tuples for voting (the order of components is of a tuple is random):

$$p_{x,i} = [ue_\gamma(f_{x,A,i})\ ue_\chi(f_{x,X,i})\ ue_y(c_{\pi(i)})\ ue_y(c'_{\pi(i)})\ ue_y(id)\ ue_y(id')]$$

(γ and χ denote public keys of RS, $y = y_1 \cdot \ldots \cdot y_k$ is the joint public key of tallying authorities).

A 6-tuples $ap_{x,i}$ used for revocation have the following form (again, the components indicated are permuted at random):

$$[ue_\gamma(f_{x,A,i})\ ue_\chi(f_{x,X,i})\ ue_y(c_{\pi(K+i)})\ ue_y(c'_{\pi(K+i)})\ ue_y(rid)\ ue_y(rid')]$$

A ballot. If a voter has chosen t (upper/lower) part of the row j, then a voting ballot has the form $v = [p_{t,j}]$, and the anti-vote contains: $av = [ap_{t,j}; p_{t,j}]$.

4.1 Voting Process

Part I: Ballot Generation Procedure
This part of the protocol is executed in interaction between BGS, Alice and application App running on her PC:

1. Alice requests $n \geq 2$ cards.
2. App sends a request for n cards to BGS (may be through an anonymous channel),
3. BGS responds for each request with the following data :
   – voting cards,
   – commitments to the identifiers contained in these cards,
   – tokens $A, \overline{A}$,
   – commitments to the components s_1, s_2 of the cyclic shifts s used.
   – digital signature under all these data.
4. Alice obtains information about the cyclic shifts (s) and components of cyclic shifts (s_1, s_2) used in cards, T_A and the identifier id. Those information is sent through a channel that is inaccessible to the PC running App (e.g. phone, SMS, ...).
5. Alice chooses a card i for voting.
6. Alice together with App generates the deterministic signature under publicly known value z.
7. App sends signature $\text{sign}_{Alice}(z)$ - this signature determines uniquely which shift component should by revealed,
8. BGS sends RSA-RE signatures to the card chosen for voting, data for opening the ciphertexts in other cards and checking the corresponding commitments, the shift component from the voting card indicated by $\text{sign}_{Alice}(z)$ and data for checking its commitment.
9. App checks if received values satisfy commitments and displays to Alice the plaintexts and the requested shift component of the card chosen for voting.

Part II: Vote Casting

1. Alice makes her choice: she chooses a part t (upper/lower) of a row w from the card, according to the cyclic shift used and her voting preferences. Namely, if she chooses the jth candidate, then the row w has to contain a ciphertext of c_j in the voting card, that is $\pi(w) = j$.
2. App performs the following steps:

   (2.1) it creates a ballot by selecting $v = [p_{t,w}]$ and revoking-ballot $av = [ap_{t,w}; p_{t,w}]$,
   (2.2) App modifies the values contained in v and av by re-encrypting every ciphertext contained in it.
   (2.3) App contacts RS and obtains a challenge r.
   (2.4) Alice signs the challenge obtaining $\text{sign}_{Alice}(r)$, a deterministic signature scheme is used.
   (2.5) App derives (in a deterministic way) an encryption key $K := \mathbf{R}(\text{sign}_{Alice}(r))$ with a pseudorandom generator $\mathbf{R}$.
   (2.6) Alice signs v and $Enc_K(av)$; then App sends these data together with the signatures created to RS.
3. RS checks the signatures and the correctness of v. Then v is stored in the set of votes cast on the public Bulletin Board, the second (encrypted) packet is stored in a repository of revocation codes. RS also provides a receipt for Alice which is a signature of RS under both packets.
4. Alice checks, with the help of Bulletin Board, if $ue_\gamma(f_{t,A,i})$ sent with her vote agrees with the value indicated by T_A known to her.

Part III: Vote revocation. A voter can cancel his previous vote by signing a challenge which was used during the previous vote casting. The procedure of voting for the second (third, ...) time contains an additional step. It is a fair exchange between a commission and a voter. The voter sends a signature for the challenge used during the previous vote generation to the commission. The commission uses it for decryption of the revocation vote, posts it on Bulletin Board and increments the number of the canceled votes.

Before accepting a revocation vote, RS checks if plaintexts of $ue_\chi(f_{t,X,i})$ contained in vote v and revoking-vote av are the same (this prohibits canceling a vote for a different candidate).

Part IV: Tallying process. After closing the polling stations the RS servers are closed as well. From every vote v and revoking-vote av, RS takes the ciphertexts containing voting options and identifiers and sends them to the first tallying authority.

Now, the mixing procedure is executed by an array of mix servers run by independent tallying authorities. For $1 \leq i \leq \lambda$, the ith tallying authority runs a server that executes the following steps:

– it reads the RSA-RE ciphertexts from Bulletin Board $i - 1$ and checks the signatures of BGS,

– it partially decodes the ciphertexts in each block with its private key,

- it re-encrypts each ciphertext,
- it permutes the ciphertexts at random and posts them on Bulletin Board i.

The last tallying authority gets, after decryption, plaintexts of the ciphertexts included in the ballots. It presents them in the Bulletin Board λ together with a Zero Knowledge Proof of correct decoding. Now, on Bulletin Board λ one can see election results.

Part V: Vote Counting. It is checked that there are exactly two occurrences of each identifier r. Then, one counts number of votes for every candidate, subtracts the number of anti-votes and divides by 2.

5 Final Remarks

Communication Complexity. A voting card contains N positions of candidates or voting options, so there are N parts (upper/lower, vote/anti-vote). Each of those parts contains 4-tuples $(id, id', c_i, c_i^{'})$ of RSA-RE-onions and two encrypted marks $A/\overline{A}, X/\overline{X}$, hence together 18 ciphertexts. There are $4K$ parts, so there are about $100N$ ciphertexts. If 1024-bit encryption is used, then the total size of the card is about $10N$ kB, so for $K = 20$ it is about 200 kB.

Social Aspects. We have presented an Internet voting scheme which is coercion-free without assumption of certified software on a secure machine used in a voting process. As far as we know this is the only solution so far that works without any trusted element. The main weak point is quite complex design incomprehensible for an average voter, so social acceptance might be a problem.

References

1. Chaum, D.: Secret-Ballot Receipts and Transparent Integrity. Better and less-costly Electronic Voting and Polling Places. In: IEEE S&P'04
2. Chaum, D.: Punch Scan. In: Workshop on Frontiers in Electronic Elections (2005), http://www.punchscan.org
3. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. CACM 24(2), 84–88 (1981)
4. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
5. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
6. Golle, P.: Reputable Mix Networks. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 51–62. Springer, Heidelberg (2005)
7. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal Re-encryption for Mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004)
8. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In:WPES 2005, pp. 61–70 (2005)
9. Hirth, M.: Receipt-Free K-out-of-L Voting based on ElGamal Encryption. In: Workshop on Frontiers in Electronic Elections (2005)

10. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In: USENIX Security Symposium 2002, pp.339–353 (2002)
11. Klonowski, M., Kutyłowski, M., Lauks, A., Zagórski, F.: Universal Re-encryption of Signatures and Controlling Anonymous Information Flow. In: Wartacrypt (2004)
12. Klonowski, M., Kutyłowski, M., Lauks, A., Zagórski, F.: A Practical Voting Scheme with Receipts. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 380–393. Springer, Heidelberg (2005)
13. Lee, B., Kim, K.: Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
14. Moran, T., Naor, M.: Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
15. Neff, C.A.: A Verifiable Secret Shuffle and its Application to E-Voting. In: ACM Conference on Computer and Communications Security, pp. 116–125 (2001)
16. Rivest, L.R.: The Three Ballot voting system. http://theory.csail.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf
17. Smith, W.D.: Cryptography Meets Voting. http://www.math.temple.edu/~wds/homepage/cryptovot.pdf

Enforcement of Integrated Security Policy in Trusted Operating Systems

Hyung Chan Kim[1], R.S. Ramakrishna[1], Wook Shin[2], and Kouichi Sakurai[3]

[1] Department of Information and Communications,
Gwangju Institute of Science and Technology,
Gwangju 500-712, Rep. of Korea
{kimhc,rsr}@gist.ac.kr

[2] Department of Computer Science,
University of Illinois at Urbana-Champaign,
IL 61801, USA
wookshin@uiuc.edu

[3] Faculty of Computer Science and Communication Engineering
Kyushu University, Fukuoka 819-0395, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. The main focus of Trusted Operating System (TOS) research these days is on the enhanced access control of reference monitors which, in turn, control the individual operations on a given access instance. However, many real-life runtime attacks involve behavioral semantics. It is desirable, therefore, to enforce an *integrated security policy* that includes both behavioral security and access control policies. We have proposed an extended reference monitor to support both access and behavior controls. This results in a sequence of operations which is also of concern in security enforcement. This paper presents the design of the extended reference monitor for integrated policy enforcement and describes its implementation in Linux operating systems.

1 Introduction

The design and implementation of TOSs have received due attention as the OS is the last line of IT defense. A TOS includes a security kernel with which attacks by illegal access operations are adequately countered. Beyond the Discretionary Access Control (DAC) – the default in Unix – most efforts concentrate on the enforcement of enhanced non-discretionary access controls such as Mandatory Access Control (MAC), or Role-Based Access Control (RBAC).

Although conventional access control schemes support confidentiality and integrity acknowledging the necessary organizational security policy, there is an evident inadequacy. This stems from the fact that the access control mechanism makes decisions with the precision of a single access operation. The well-known example is the set-user-id (setuid) mechanism in generic Unix or Linux OS. Usually, a privilege transition is needed when a user has to perform a special task such as changing the password, configuring the system, or enabling/disabling

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 214–229, 2007.
© Springer-Verlag Berlin Heidelberg 2007

hardware devices. Security administrators insist on safe execution of such programs, i.e., bounded execution by operations defined (compiled) in the program binary. However, we have constantly witnessed runtime bugs in such programs from security advisories. If the program is subverted by Stack Overflow attack [1], for example, the attacker can hold the transferred privilege of the program; and this is clearly not the intention of the administrators. If we carefully observe the attack process, we find that the underlying DAC access control context is valid even though the attack has succeeded. This subversion relies on *the change of behavior* of the program under legitimate access control context. The situation with TOS is no different if it (the TOS) includes the privilege transition mechanism in its access control scheme for whatever reason. This deficiency is due to the lack of another dimension in security enforcement besides access control.

We have proposed an extended reference monitor [2,3] with a view to incorporate the behavioral dimension in security enforcement. A TOS with the extended reference monitor can enforce an integrated security policy that manages not only a single access operation but also a sequence of consecutive operations. We find that the control of a relation of operations (an operation sequence) by a reference monitor is indeed *behavior control* as opposed to the usual *access control*.

This paper proposes the design of the extended reference monitor that attempts to enforce an integrated security policy in TOSs. The fundamental structure is based on Domain and Type Enforcement (DTE) [5]. DTE facilitates efficient access control configuration for a TOS. It is also very simple and highly flexible in managing access entities. As for access control, we provide Role-Based Access Control (RBAC) [6]. The incorporation of RBAC into DTE provides much needed flexibility as further abstraction of subjects is possible with RBAC. We borrow the concept of a fluent from Event Calculus (EC) [7]. With the help of fluents, we can express the behavioral constraints of processes which go a long way in behavior control. We also describe the design of an integrated policy language. An implementation of the extended reference monitor in Linux kernel has also been provided.

The rest of this paper is organized as follows: In section 2, the motivation for extending the reference monitor is indicated. Section 3 describes the protection structure of the extended reference monitor that is relevant in enforcing an integrated policy. Section 4 presents the design of the policy language. Implementation of the reference monitor is described in section 5. Section 6 summarizes related work. We conclude this paper in section 7.

2 The Motives of Extended Reference Monitor

We encounter various types of program attacks on OSs on a regular basis. Attacks such as Stack and Heap Overflow, Double Free, and Format String accompany shell code aiming to mess up the privilege beyond the currently operative rights of users. Attacks exploiting race condition of some processes also lead to security hazards. We note that these attacks are related to the two functions described below.

The first function is the privilege transition that is very helpful in system management. Generic UNIX involves DAC and the *setuid* is the mechanism for privilege transition. As for TOSs whose access control functions go beyond DAC, researchers tend to adopt analogous mechanisms in their implementations. GRSecurity's role transition [14] and SELinux's domain transition [9] provide examples of the trend. With these mechanisms, extracting privileges out of one's duty is possible if a program contains a bug that becomes active during program execution. None of the access contexts of the attack sequence violates the underlying DAC enforcement rules. For example, each individual access operation is legal in a typical Stack Overflow attack. The behavior encoded in the program binary is replaced with the behavior of the attacker starting from the shell code, and the latter violates security requirements. As the privilege mode is transited by a successful attack, the reference monitor cannot spot the subject of the compromised process [17]. For more details on this situation, see [2]. If a TOS includes any such privilege transition and contains a bug, then a similar situation obtains: we have investigated such a possibility in SELinux [8].

The other function is process concurrency. Needless to say, it is a very important feature in OSs. However, there exists a set of attacks which exploit this feature, i.e., time-of-check-to-time-to-use (TOCTTOU) attacks [4]. The attack is caused by the programmer's dubious assumption that between two consecutive operations – one checking a resource and the following one using the resource – the reference of the resource does not change. This assumption is not true and the attackers have exploited this misunderstanding. As in the case of stack overflow, access contexts of involved operations do not violate the given access control enforcement rules. The problem arises because of the specific sequence (behavior) of operations, the effect of scheduling made by the attacker, of the involving processes.

The above situation arises because access control with conventional reference monitors does not consider the (semantic) relation among consecutive access operations, although many real-life program runtime attacks do involve such behavioral semantics. The hazard is that even though all the invoked access operations are legal individually, the sequence of operations may result in an attack. Therefore, without behavior control, a conventional reference monitor can not counter real life attacks effectively even though the monitor implements correct access control.

We have proposed an extended reference monitor [2,3] with a view to involve the behavioral dimension in security enforcement. Our monitor model attaches a behavior control component to the access control unit, thereby providing both access and behavior controls in the system. In [2], we have defined the behavioral control unit as a *security automata* [16] which can guarantee the *safety property* which ensures that if a trace is not in accordance with the behavioral policy, then the execution must be terminated at the earliest. We have extended the RBAC model [3] with the procedural restriction; thus, the monitor makes access decisions based on continuous access information in RBAC-enabled TOSs. This paper modifies the extant protection model [2] and also describes a new policy language implementation.

3 Protection Structure for the Integrated Security Policy

In this section, we propose a protection structure model for the extended reference monitor in order to realize security enforcement under an integrated policy.

In any OS, the user invokes an operation on a target object within a process context. We allow basic system entities to be users, operations, objects, and processes. In accordance with the DTE, processes and objects are labeled by domain and type respectively. We define an *event* as an invocation of an operation on an object that is labeled with a certain type. Operation-type-object (object) semantic is created when an event occurs [Fig. 1]. Here is how the entities are defined:

- $USER, OPR, OBJ$: the set of users, operations and objects.
- $PROC$: the set of processes.
- $exec(u : USER) \rightarrow 2^{PROC}$: executing processes activated by a user u.
- $DOMAIN, TYPE$: the set of domains and types.
- $d\_label(p : PROC) \rightarrow DOMAIN$: domain label of a process p.
- $t\_label(ob : OBJ) \rightarrow TYPE$: type label of an object ob.
- $EVENT \subseteq OPR \times TYPE$: the set of events.

We support two types of policy: *open* or *closed*. An open policy specifies prohibited events. The enforcement with an open policy initially allows all the (access) events and restricts events which are specified as rules. A closed policy, on the contrary, specifies allowable events, initially prohibiting all the events. Usually, the enforcement of a closed type policy in an OS incurs considerable performance penalty: the tightly regimented processes under a closed policy need to be checked for all the operations, while the reference monitor checks only the concerned operations with open policies. The access control enforcement in an OS can be open or closed. However, the behavior control enforcement with a closed policy may lead to a slower system (due to state management). Thus it might be better to enforce behavior control with open policies. However, if an administrator wants to make a process – which maybe introduced into the systems for the first time (thus, untrustworthy) – to be tightly regimented, recourse may be taken to a closed behavior policy. The other reason is that the policy expression for behavior control is easy with an open policy type rather

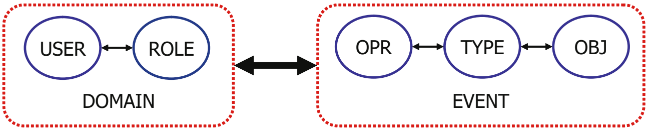
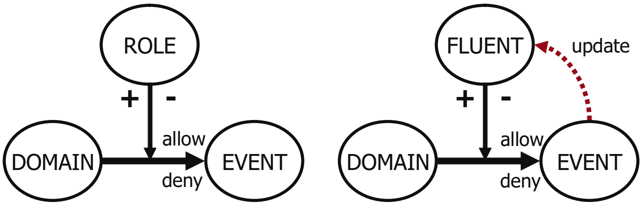


Fig. 1. Our protection structure for the extended reference monitor



(a) Access control decision (b) Behavior control decision

Fig. 2. Security decisions

than with a closed one. The ordinary host based intrusion detection system checks the condition for attacks and denies access if the condition is detected.

Each domain is under an open or a closed policy.

- $POLICY\_TYPE = \{open, closed\}$: there are two types of policy.
- $policy\_type(d : DOMAIN) \rightarrow POLICY\_TYPE$: policy type of a domain d.

For the access control part of an integrated policy, we rely on the RBAC. In an OS, a user executes an executable and the execution instance creates a user-role-domain (subject) semantic [Fig. 1]. This means that a program process (being executed by some user), and access operations of the program are controlled by the eligibility of the role. A user can be assigned to a set of *roles* in order to execute relevant job functions.

Access control is implemented in such a way that a user in a subject semantic can produce an event by *eligibility of user's role* in the event [Fig. 2(a)]. The reference condition on eligibility may be positive (+) or negative (−). Positive eligibility of a role leads to the condition: "when a user executing a program of the domain is assigned to the role." In contrast, negative eligibility refers to the condition: "when a user executing a program of the domain is not assigned to the role." The access decision function (adf) checks the conditions. Note that the *adf* does not compute the final decision.

- $ROLE$: the set of roles
- $UA \subseteq USER \times ROLE$: a many-to-many user-to-role relation.
- $ROLE\_COND \subseteq ROLE \times \{+, -\}$: the set of role reference condition.
- $adf(u : USER, r\_cond : ROLE\_COND) \rightarrow \{true, false\}$: on user's role assignment, access decision function returns *true* in accordance with the role reference condition, i.e,
  - $(\forall u : USER, \forall r\_cond : ROLE\_COND) adf \Rightarrow (\exists r : ROLE)[((r, u) \in UA \wedge (r, +) \in r\_cond) \vee ((r, u) \notin UA \wedge (r, -) \in r\_cond)]$

We were motivated by Event Calculus (EC) [7] in configuring the behavior control policy. The concept of *fluent*, a time-variant property, is included to capture the behavioral aspect of processes. A fluent is initiated by an event thereby making the fluent *hold* (value: *true*). Conversely, an event may terminate the fluent in which case the fluent does *not hold* (value: *false*). In behavior control, fluents are defined to reflect certain states in a behavior. As the fluents are updated by concerned events [Fig. 2(b)] the reference monitor can compute a decision by considering not just a single event but a sequence of events. Behavior decision function (bdf) checks whether the fluent is *hold* (for positive condition) or *not hold* (for negative condition).

- $FLUENT$: the set of fluents.
- $f\_val(f : FLUENT) \rightarrow \{true, false\}$: the value of a fluent f.
- $FLUENT\_COND \subseteq FLUENT \times \{+, -\}$: the set of fluent reference condition.
- $bdf(f\_cond : FLUENT\_COND) \rightarrow \{true, false\}$: behavior decision function returns *true* if the given fluent reference condition is satisfied, i.e,
  - $(\forall f\_cond : FLUENT\_COND) bdf \Rightarrow (\exists f : FLUENT)[(f\_val(f) \wedge (f, +) \in f\_cond) \vee (\neg f\_val(f) \wedge (f, -) \in f\_cond)]$

Security decisions are made based on the subject and the object semantics; and, this can be achieved by relating a domain and an event with appropriate reference conditions. We define an integrated policy rule as follow:

- $RULES \subseteq DOMAIN \times EVENT \times ROLE\_COND \times FLUENT\_COND$: the set of security rules.

A rule refers to a user's restricted events (permission, i.e., operation on an object) if the domain associated with the user's executing process is under an open policy; and, a user's allowed events in the case of closed policy.

Security decision function (sdf) provides final decisions for allowing an event based on encoded rules. The function refers to *adf* and *bdf* for access and behavior control conditions. Under an open policy, if both the conditions are satisfied, then the process' event with a certain domain should be denied. On the contrary, under a closed policy, the event should be permitted.

- $sdf(u : USER, p : PROC, op : OPR, ob : OBJ) \rightarrow \{true, false\}$: On invocation of an operation op with an object ob by a user u within a process context p, security decision function returns *true* for a permitted operation and returns $false$ for a rejected operation, i.e,
  - $(\forall u : USER, \forall p : PROC, \forall op : OPR, \forall ob : OBJ) sdf \Rightarrow (\exists r\_cond : ROLE\_COND, \exists f\_cond : FLUENT\_COND)$ $[p \in exec(u) \wedge \neg((policy\_type(d\_label(p)) = open) \wedge (d\_label(p), (op, t\_label(ob)), r\_cond, f\_cond) \in RULES \wedge adf(u, r\_cond) \wedge bdf(f\_cond)) \vee ((policy\_type(d\_label(p)) = closed) \wedge (d\_label(p), (op, t\_label(ob)), r\_cond, f\_cond) \in RULES \wedge adf(u, r\_cond) \wedge bdf(f\_cond))]$

4 Security Policy Configuration

This section describes the integrated policy language for the extended reference monitor.

4.1 Declarations and Domain/Type Labeling

First of all, we need to define entities involved in a configuration before they are used. The declarations of users, roles, domains, and types are very similar to what we have been doing in variable declarations in programming languages: we place keywords *user*, *role*, *domain*, and, *type* before the name of each entity.

We may define a user *john*, a role *admin_r*, a domain *admin_config_d*, and a type *etc_passwd_t* as follows:

```
user john;
role admin_r;
domain admin_config_d;
type etc_passwd_t;
```

A domain defines a restricted execution semantic of programs. A set of programs are labeled with a specific domain with *domain_assign* keyword in the following way:

$$\textit{domain\_assign (-r) domain\_name} \rightarrow \{ \textit{ executable program paths } \};$$

The executable program path designates the exact path of a specific program without the recursive option (-r); and, with the option, directories are specified and all the executables in the directories are labeled with the domain name.

The init process executed from /sbin/init can be labeled with *init_d*, and processes invoked from executables in /sbin, /usr/sbin, /usr/local/sbin, /usr/bin/kerberos/sbin be labeled with *admin_d* as follows:

```
domain_assign init_d -> { /sbin/init };
domain_assign -r admin_d ->
{ /sbin, /usr/sbin, /usr/local/sbin, /usr/bin/kerberos/sbin };
```

Type labeling can be done in a way similar to domain labeling with *type_assign* keyword. The difference is that a type can be assigned to any file object including normal files, directories, or even devices (for example, we may assign *console_t* to control read or write operation to console).

4.2 Event and Fluent Definitions

As our policy language is influenced by the EC, defining events and fluents are crucial to policy configuration. In our TOS framework, access operations to be controlled are all defined by events. The event definition is an expression of the form:

$$\textit{event event\_name } \{ \textit{ op: operation; ob: object; arg; argument\_expression; } \};$$

The event definition is comprised of three fields: operation type (*op*), target object (*ob*), and argument expression (*arg*). The operation field must be specified, but the object and the argument fields are optional. Our TOS implementation depends on Linux Security Module (LSM) [12]; thus, the operation type is the part of operations hooked by LSM, such as *stat*, *setuid*, *execve*, and so on. The target object can be of any type on which the operation is valid. Just as system call functions take arguments, we can specify the arguments for the operation and this enhances the granularity of event control.

Example 1. The event *chroot_non_root_dir_e* specifies chroot operation on the directories except the system root directory. The *set_euid_zero_e* event expresses setuid operation with both ruid and euid arguments to be zero.

```
event chroot_non_root_dir_e { op: chroot; ob: !root_dir_t; };
event set_euid_zero_e { op: setuid; arg: ruid = 0, euid = 0; };
```

A fluent starts *to hold* by initiating events. It enters *not hold* state by terminating events. The fluent definition is of the form:

fluent fluent_name { *i*: *initiating events*; *t*: *terminating events*; (*hold/!hold*;) };

We can set the initial value of the fluent by specifying *hold* or !*hold* for initially holding and initially not holding. Without the setting, the fluent is initially in the !*hold* state. In our policy language, a fluent can have a valid range. By default, events enlisted in a fluent are limited to the local boundary: the fluent is only affected by the events within the local process. However, it is reasonable to suppose that the security of a process may be in the shadow of other processes in the OS; hence, we let the fluent be influenced by events in external domains. We include *domain_name.event_name* for this purpose. This means that the event of a specific domain can influence the fluent which includes that event (in initiating or terminating the fluent). The keyword *ext_domain* is included to define alias of a domain name for the external event definition in order to abstract domain names.

Example 2. The fluent *euid_zero_f* starts to hold when the event *set_euid_zero_e* has occurred; the event *unset_euid_zero_e* terminates the fluent. The *checked_f* can be terminated by the *use_open*1*_e* local event, and also by the event *unlink_e* which is triggered by the external domain *normal_d*.

```
fluent euid_zero_f {
    i: set_euid_zero_e;
    t: unset_euid_zero_e; };
fluent checked_f {
    i: check_stat_e;
    t: use_open1_e, attacker_d.unlink_e; };
ext_domain attacker_d -> { normal_d };
```

4.3 Policy Specification and Its Assignment

The integrated security policy consists of two parts: access and behavior control specification. We describe access control part first. We declare an access control specification to be open (*a_spec* with *open* declaration) and make a list of prohibited events with access control expressions. We declare a closed policy with *a_spec* with *close* keyword. If there is no policy type for a policy specification, it is considered to be an open policy. The access control specification has the following form:

a_spec spec_name { (*open/close*;) *access control expressions* };

Access control expression presents an allowed or a disallowed event depending on the role which is assigned to the user who is executing a process of some domain. The check for role assignment can be positive (+) or negative (-): the positive condition checks for the assignment of the role, and the negative condition checks for the negation of role assignment. Access control expression has the following form:

(+/-) *role_name* → *allow/deny event_name*;

One can incorporate access control specifications to some domains with the *a_assign* definition.

Example 3. The specification *no_access_to_some_config_file* implies that a user who is executing programs of *normal_d*, or *ftp_d* cannot trigger the event *open_http_config_e* if the user is assigned to the role *user_r*. Also the user cannot trigger the event *open_php_config_e*, if he or she is not assigned to *admin_r*.

```
a_spec no_access_to_some_config_file {
    +user_r -> deny open_http_config_e;
    -admin_r -> deny open_php_config_e; };
a_assign no_access_to_some_config_files -> { normal_d, ftp_d };
```

The main entities of behavior control specification are event and fluent. When an event is about to occur, it is disallowed or allowed based on the involved fluents' state; and the states may be affected by the event. The syntax for behavior control specification is very similar to the access control specification. The behavior control specification is of the form:

b_spec spec_name { (*open/close*;) *involving fluents*;
behavior control expressions };

A behavior control expressions has the following form:

(+/-) *fluent_name* → *allow/deny event_name*;

For the positive case, the expression is interpreted in terms of the *holdsat* predicate in the EC; therefore, it means that if a fluent is in the hold state, then the event is allowed/disallowed. The negative expression is interpreted as ¬*holdsat*.

Example 4. The specification *prevent_chroot_to_root_dir* means that if the fluent *chroot_ non_root_dir_invoked_f* is in the hold state, then the event *chroot_root_dir_e* must not be allowed.

```
b_spec prevent_chroot_to_root_dir {
    fluent chroot_non_root_dir_invoked_f;
    +chroot_non_root_dir_invoked_f -> deny chroot_root_dir_e; };
b_assign prevent_chroot_to_root_dir -> { normal_d };
```

4.4 Behavior Specification Examples

Configuring access control with our policy is straightforward. Here we describe how the EC based behavior specification can be expressed to reflect real life examples of attack.

A runtime attack such as Buffer Overflow tries to change the execution flow in the user mode. If the attacker succeeds, he/she usually spawns a shell in the privileged mode. The shell code usually contains a sequence of $setreuid(0, 0)$ followed by $execve(SHELL)$ system calls. The following example configuration monitors the behavior. The fluent *euid_zero_f* checks the status of effective uid (euid) which is set by the family of $set*uid$ system calls; the execution of a shell is disallowed if the fluent is in the hold state. If the process is not affected by the attacks, and if the privileged mode is normally exited, the fluent is flipped so that we can restore its first state.

Example 5. No shell execution after setting the euid to be zero.

```
event set_euid_zero_e { op: setuid; arg: ruid = 0, euid = 0; };
event unset_euid_zero_e { op: setuid; arg: ruid != 0, euid != 0; };
event shell_exec_e { op: execve; ob: shell_t; };

fluent euid_zero_f {
    i: set_euid_zero_e;
    t: unset_euid_zero_e; };
b_spec no_shell_when_ruid_and_euid_zero {
    fluent euid_zero_f;
    +euid_zero_f -> deny shell_exec_e; };
b_assign no_shell_when_ruid_and_euid_zero -> { normal_d, http_d };
```

The following example configuration prevents a TOCTTOU attack: i.e. the file race condition. With this policy, the monitor prohibits the time-of-use if *unlink* and *symlink* operations are invoked on files labeled *tmp_t* followed by the time-of-check. The value of involved fluents is affected by both local and external processes thereby achieving adequate monitoring.

Example 6. Preventing TOCTTOU attack.

```
event check_stat_e { op: stat; ob: tmp_t; };
event use_open1_e { op: access; ob: tmp_t; };  # normal use
event use_open2_e { op: access; ob: !tmp_t; }; # abnormal use
event unlink_e { op: unlink; ob: tmp_t; };
event symlink_e { op: symlink; ob: config_t; };

fluent checked_f {
    i: check_stat_e;
    t: use_open1_e, attacker_d.unlink_e; };
fluent unlinked_f {
    i: attacker_d.unlink_e;
    t: use_open1_e, attacker_d.symlink_e; };
fluent symlinked_f {
    i: attacker_d.symlink_e;
    t: use_open1_e; };

b_spec tocttou {
    fluent checked_f, unlinked_f, symlinked_f;
    +symlinked_f -> deny use_open2_e; };
b_assign tocttou -> { admin_config_d };
ext_domain attacker_d -> { normal_d };
```

5 Implementation

Here we present a proof-of-concept implementation of the integrated security policy enforcement.

5.1 Policy Compiler

We have implemented a policy compiler and some utilities for the extended reference monitor. The compiler has been developed in the Python version of Lex/Yacc module [28]. It parses access and behavior control specifications (integrated policy) and generates a policy binary. The binary is loaded into the Linux kernel module which implements the extended reference monitor. The domain and the type labeling utilities are also built in order to appropriately label domains and types for numerous access entities, especially file objects, in the system.

5.2 Extended Reference Monitor

We have patched Linux kernel version 2.6.17 to include our extended reference monitor functionality. Our system architecture follows the general enforcement

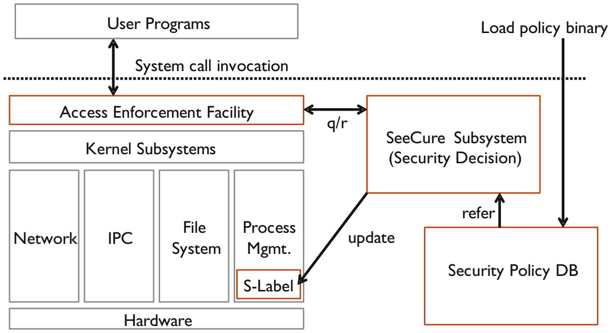


Fig. 3. System architecture

structure [Fig. 3] to clearly discriminate between security enforcement and decision facilities [10,11]. We have adopted Linux Security Module (LSM) [12] as access enforcement facility (AEF) which redirects the system call invocation to Security Decision Subsystem (SDS), the implementation of security decision function *sdf* which is shown in section 3. The query for security decision is made by the SDS which refers to Security Policy DB (SPDB), which contains the integrated security policy that governs access and behavior control configuration built by the policy compiler.

The access control information (ACI) for objects is accompanied by the object itself. For example, the type information for file object is labeled in the extended attributes of files which are supported by EXT3 filesystem. The ACI for the subject is encoded in the process management subsystem (*task_struct* structure in Linux process definition) as a security label (S-Label) wherein the role and fluent information is labeled in the process creation stage (referring the SPDB). At decision time, SDS makes security decisions from *sdf* based on those ACIs.

5.3 Performance Evaluation

We have conducted performance evaluation tests on our implementation on a 2.4 GHz PC with 1GB memory. The kernel level measurement of the system call [1] shows that 4.7% overhead on process creation involves $fork$ and *execve* system calls. The overhead may be incurred due to security labeling of subjects.

When defining an event, we can include objects and argument expressions for an operation and those can influence the system call performance. The overhead due to *mmap* system call without specifying anything in the event definition was 1.17%. With a target object specification, the overhead due to *stat* was 4.3%. The overhead due to *setreuid* with full argument specification was 12.6%. Using the UnixBench [27] benchmark test tool, we have measured the file copy overhead (which involve $permission$ routine in LSM, and the routine is frequently used in system calls which handle files) with 256, 1024, and 4096 bytes; and the results were 6.2%, 3.7%, and 4.7%, respectively.

In our system, processes not associated with policies are not affected by the enforcement. Here we present the performance of a process that is tightly regimented: i.e., all the operations are checked by the SDS. It is meaningful to consider it as it shows an upper bound on the performance of the process. Figure 4 shows the result of comparison between the original kernel and the patched kernel. Each kernel executes a simple program with a closed policy. We measured the execution time of the program; and the result shows that the patched kernel has a small overhead.

It is natural that the decision time would increase with the number of fluents in a behavior policy. We also observed that the computational cost of a *setreuid* system call with increasing number of fluents and the result is depicted in Figure 5. The numbers in the graph specify time in nanoseconds. It appears

[1] We use the process's time stamp counter (TSC) for the measurement of cycles and refer the cycles to time conversion routine in timer_tsc.c in Linux kernel source tree.

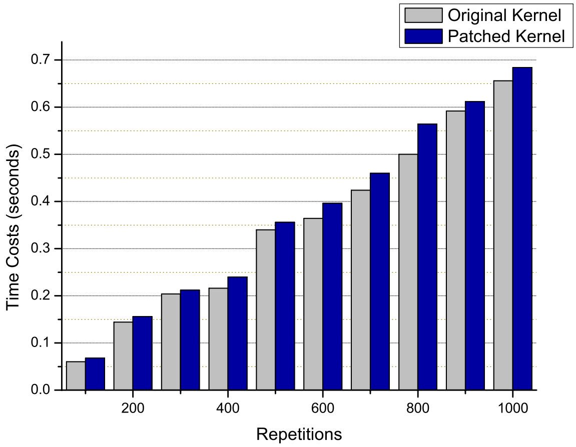


Fig. 4. Comparison of the execution time between the original kernel and the patched kernel

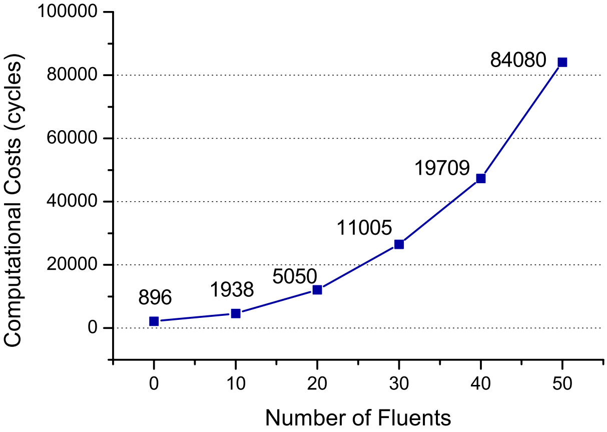


Fig. 5. Performance of security decision: the computational cost is the instruction cycles measured with rdtscll registers, and the numbers on the curve indicate time in nanosecond

that a load of more than 40 fluents in a process brings down the performance. However, we use policies with a view to invoke the concerned operations. Real enforcement is expected to be quite different: the number of behavior policies is not very large, in general. Access control covers a large part of the security enforcement in operating systems; and a small number of structured behavior policies would be enough to cope with a class of attacks.

The main objective of our implementation is to investigate the operativeness of the extended reference monitor functionality. Current implementation does not aim at optimized performance. Inclusion of a cache mechanism for access rules appears to be promising in this regard.

6 Related Work

There are many TOS projects that employ kernel-based reference monitors. The architecture of Rule-Set Based Access Control (RSBAC) for Linux Kernel [13] is based on Generalized Framework for Access Control (GFAC) and it supports multiple access control policies, especially MAC. The Security-Enhanced Linux (SELinux) [9] implements the Flask architecture to support several access control policies with a high degree of flexibility. The major access control schemes of SELinux are RBAC and DTE (Domain and Type Enforcement). Both RSBAC and SELinux make a clear separation of enforcement from decision facility; and these architectures are now commonly incorporated. REMUS [15] also has similar enforcement architecture, but its access control mechanism is based on a detailed analysis and empirical classification of Linux system calls. RSBAC, SELinux, and REMUS make access decisions on a given single access solely through access control policy; and this point sets our approach apart. We also incorporate behavior as well as a given single access operation for decisions. Thus we enforce integrated security policies with the extended reference monitor.

On the other hand, there are also efforts that constrain the behavior by specifying a behavioral policy. Linux Intrusion Detection System (LIDS) [18] and Blue Box [19] are policy-driven intrusion detection systems in the kernel layer. Both can specify the policy about behavior based on process' capability. However, their specification model is based on an empirical approach rather than a formal one. SASI (Security Automata SFI Implementation) [21] is a reference monitor framework that enforces security policies based on Security Automata. Its implementation layer is at a lower level than ours as their target operation unit is x86 machine instruction. The intrusion detection based on the specification language [20] also describes behavioral policy to detect the unknown behavior of applications. The above systems do not provide a provision for the enhanced access control schemes such as MAC or RBAC.

Zimmermann et al. presented very similar observations in respect of deficiencies of conventional reference monitor [17]. However, their approach is different from ours. They adopted the *information flow* concept in security enforcement. Although their monitor considers relation of consecutive actions, it checks information flows between objects rather than explicitly constraining each operation.

Other than TOS area, there is some work on access control based on the EC. Efstratiou et al. have developed a policy language based on simple EC for mobile adaptive systems [23]. While their work is on the implementation of a specific system, Bandra et al. have presented a reasoning framework for general security policies [24]. Baker [22] has applied the simplified EC to RBAC. Tishkov et al. presented an architecture for access control policy management mainly dealing with RBAC [25].

In our implementation, we have used the LSM [12] to be AEF: i.e. hooking security-relevant operations of kernel. Many kernelized monitors are implemented on this because it supports generic hooking method replacing the system call interception. SELinux represents work of the LSM type. Early versions of RSBAC [13] and GRSecurity [14] were also on top of the LSM. Jaeger et al. have worked on the verification of hook placement of the LSM [26].

7 Conclusions

In this paper, we have proposed a protection model for the enforcement of an integrated security policy in TOSs. Our protection structure has attempted to integrate DTE with RBAC. The monitor checks the eligibility conditions while controlling the access. We adopted the fluent concept to effectively enforce the behavioral policy. Various behavioral features of real-life runtime attacks can be configured in this way. Our reference monitor can accommodate behavioral dimension of the security policy. A policy language suitable for security configuration has been described with examples that show its effectiveness. We have also described the proof-of-concept implementation of the reference monitor in the Linux kernel. We believe that our enforcement of an integrated security policy enhances the operativeness of conventional access control systems.

Acknowlegements. The authors gratefully acknowledge the reviewer's helpful comments in preparing the final manuscript. This research was supported in part by the BK21 of Ministry of Education of Korea, and in part by Strategic International Cooperative Program, Japan Science and Technology Agency (JST), of Japan.

References

1. Alphe One. Smashing the stack for fun and profit. Phrack Magazine 7 (49), File 14 of 16 (1996)
2. Kim, H.C., Shin, W., Ramakrishna, R.S., Sakurai, K.: Design and implementation of an extended reference monitor for trusted operating systems. In: Chen, K., Deng, R., Lai, X., Zhou, J. (eds.) ISPEC 2006. LNCS, vol. 3903, pp. 235–247. Springer, Heidelberg (2006)
3. Shin, W., Park, J.Y., Lee, D.I.: Extended role based access control with procedural constraints for trusted operating systems. IEICE Trans. Inf. & Syst. E88-D(3), 619–627 (2005)
4. Lowery, J.C.: A Tour of TOCTTOUs. SANS GSEC practical v.1.4b (August 2002)
5. Badger, L., Sterne, D.F., Sherman, D.L., Walker, K.M., Haghighat, S.A.: A domain and type enforcement unix prototype. In: Proc. of the 5th USENIX UNIX Security Symposium (1995)
6. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Chandramouli, R.: Role-based access control models. IEEE Comput. 29(2), 38–47 (1996)
7. Miller, R., Shanahan, M.: Some Alternative Formulations of the Event Calculus. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS (LNAI), vol. 2408, pp. 452–490. Springer, Heidelberg (2002)

8. Kim, H.C., et al.: On the privilege transitional attack in secure operating systems. In: CSS2004. Proc. of Computer Security Symposium 2004, vol. II, pp. 559–564 (2004)
9. Loscocco, P., Smalley, S.: Integrating flexible support for security policies into the linux operating system. In: USENIX Annual Tech. Conf. (2001)
10. Abrams, M.D., LaPadula, L.J., Eggers, K.W., Olson, I.M.: A generalized framework for access control: An informal description. In: Proc. of the 13th Nat'l Comput. Sec. Conf. pp. 135–143 (1990)
11. Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., Lepreau, J.: The flask security architecture: system support for diverse security policies. In: Proc. of The 8th USENIX Sec. Symp. pp. 123–139 (1999)
12. Wright, C., Cowan, C., Morris, J., Smalley, S., KroahHartman, G.: Linux Security Modules: General Security Support for the Linux Kernel. In: Proc. of USENIX Security Symposium (2002)
13. Ott, A.: The rule set based access control linux kernel security extension. In: Int'l Linux Kongress 2001 (2001), http://www.rsbac.org
14. Spengler, B.: Increasing performance and granularity in role-based access control systems (A case study in Grsecurity), http://www.grsecurity.net/
15. Bernaschi, M., Gabrielli, E., Mancini, L.V.: REMUS: A security-enhanced operating system. ACM Trans. on Inf. & Syst. Sec. 5(1), 36–61 (2002)
16. Schneider, F.B.: Enforceable security policies. ACM Trans. on Inf. & Syst. Sec. 3(1), 30–50 (2000)
17. Zimmermann, J., Mé, L., Bidan, C.: An improved reference flow control model for policy-based intrusion detection. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 291–308. Springer, Heidelberg (2003)
18. Linux intrusion detection system. http://www.lids.org/
19. Chari, S.N., Cheng, P.: BlueBox: A policy-driven, host-based intrusion detection system. ACM Trans. on Inf. & Syst. Sec. 6(2), 173–200 (2003)
20. Sekar, R., Bowen, T., Segal, M.: On preventing intrusions by process behavior monitoring. In: Proc. of Workshop on Intrusion Detection and Network Monitoring, pp. 29–40 (1999)
21. Erlingsson, U., Schenider, F.B.: SASI enforcement of security policies: a retrospective. In: Proc. of the New Security Paradigm Workshop, pp. 87–95 (1999)
22. Baker, S.: Data protection by logic programming. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 1300–1314. Springer, Heidelberg (2000)
23. Efstratiou, C., Friday, A., Davies, N., Cheverst, K.: Utilising the event calculus for policy driven adaptation on mobile systems. In:(Policy 2002) Proc. of the 3rd Int'l Workshop on Policies for Distributed Systems and Networks (2002)
24. Bandara, A.K., Lupu, E.C., Russo, A.: Using event calculus to formalise policy specification and analysis. In:(Policy 2003). Proc. of the 4th Int'l Workshop on Policies for Distributed Systems and Networks (2003)
25. Tishkov, A., Kotenko, I., Sidelnikova, E.: Security checker architecture for policy-based security management. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 460–465. Springer, Heidelberg (2005)
26. Jaeger, T., Edwards, A., Zhang, X.: Consistency analysis of authorization hook placement in the linux security modules framework. ACM Trans. on Info. and Syst. Sec. 7(2), 175–205 (2004)
27. UnixBench. http://www.tux.org/pub/tux/benchmarks/System/unixbench/
28. PLY. http://www.dabeaz.com/ply/

Salvia: A Privacy-Aware Operating System for Prevention of Data Leakage

Kazuhisa Suzuki[1], Koichi Mouri[2], and Eiji Okubo[2]

[1] Graduate School of Science and Engineering, Ritsumeikan University, Japan
[2] College of Information Science and Engineering, Ritsumeikan University, Japan

Abstract. We have developed a privacy-aware operating system that focuses on preventing leakage of sensitive data such as personal information. The existing mandatory access control model is too restrictive for processes required to sustain the operations of user programs such as FTP, e-mail client applications, etc. In order to solve this problem, the proposed approach employs two techniques. First, the operating system kernel limits the execution of system calls only if the process could contribute to data leakage. Second, we implemented contexts; contexts are parameter or hints facilitating the evaluation of the risk of data leakage. These contexts also determine whether the kernel allows or disallows the execution of system calls. These techniques make it possible to realize a more adaptive and flexible data protection mechanism than the existing ones. This study describes the proposed approach.

1 Introduction

Recently, several incidents of information leakage have occurred making them a serious problem. In particular, if the leaked information contains sensitive data such as a company's confidential information or private data (name, telephone numbers, e-mail addresses, credit card numbers, etc.), the threat of leakage becomes even more dangerous. These incidents could not only result in fraudulent or fictitious claims but also a loss of trust in the company associated with the incidents. A report on information leakage incidents in Japan (by National Consumer Affairs Center of Japan) lists the following primary reasons for the occurrence of the information leakage: a member or an operator of a company leaks data accidentally, a member or an operator of a company contributes to the leakage of data by carrying mobile computers or portable storage devices outside the company, theft of computers or storage devices, or illegal intrusion into network servers due to misconfiguration.

A file is the basic unit of data storage in conventional operating systems and most operating systems provide file systems to manage files. One of the most important functions of a file system is file protection. Discretionary access control (DAC) is a popular access control model for realizing file protection. DAC is used to limit users' access to a file based on the file's permissions. However, DAC is unaware of the purpose for which a process accesses a file. Thus, it is impossible to distinguish between a proper file access and a malicious one. Further, it is

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 230–245, 2007.
© Springer-Verlag Berlin Heidelberg 2007

impossible to prevent a process from accessing other files, pipes, and/or sockets. On the other hand, mandatory access control (MAC) is a fine-grained access control model that has been adopted by certain trusted operating systems to protect computation resources. MAC enforces access restrictions on all processes or users regardless of their privileges. The Bell-LaPadula Model [1], which is one of the most famous access control models for MAC, and the other models for MAC, impose restrictions that are too strict to sustain user processes.

Typically, users want their private data to be used by data administrators or operators under an agreement such as the "Privacy Policy Declaration" of a company. Therefore, it is necessary to be able to restrict the purposes of data manipulation and the scope of data distribution according to an agreement.

In this study, we address cases wherein a person with valid access privileges to sensitive data contributes to data leakage and propose an operating-system-level approach for the prevention of data leakage. We have developed *Salvia* – a privacy-aware operating system – to address the abovementioned shortcomings of DAC and MAC and satisfy the requirements of private data owners. *Salvia* employs an adaptive data protection mechanism that is suitable for preventing data leakage. The remainder of this paper is organized as follows: Sect. 2 describes the shortcomings of DAC and MAC that need to be addressed to prevent data leakage; Sect. 3 presents *Salvia*'s data protection mechanism, especially the specifications, and describes how to configure data protection policies with contexts; Sect. 4 provides an overview of *Salvia*'s implementation; Sect. 5 discusses the evaluation of *Salvia*; Sect. 6 presents related works; and Sect. 7 presents the conclusion.

2 Background

We describe two (example) scenarios to discuss the shortcomings of DAC and MAC in preventing data leakage. In the first scenario, an operator who manipulates customers' private data unintentionally attaches a file containing private data to an e-mail and unaware of the fact that the file is attached he/she sends the e-mail to someone. Thus, in this scenario, the operator's mistake leads to data leakage.

DAC is not suitable for protecting a file against being sent via the e-mail. If operators are denied access to a file (no read permissions), they cannot accidentally attach the file to an e-mail, but neither can they perform tasks that involve the use of the file. If they are denied access to the e-mail program, they cannot send/receive any e-mails, even when necessary.

In MAC, security labels or classifications are assigned to each computation resrouce, and the access requests of only those processes and/or users are allowed who possess the required levels of authorization or clearance. In order to clarify the shortcomings of MAC in this scenario, we consider several additional conditions: The file that contains customers' personal information and needs to be protected is stored on a hard disk drive as a plaintext file. The operator uses a text editor such as GNU Emacs to edit the file. If GNU Emacs has read permission for the file and has all permissions for the TCP sockets, the file may be leaked via an e-mail program executed from GNU Emacs. If GNU Emacs does not have any permissions for the TCP sockets, the operator cannot send any e-mails. This pattern is similar to the one posed by the way of DAC.

In the second scenario, an operator carries a mobile computer containing a confidential file outside the company against the company's policy. The operator's act leads to data leakage in this case. Even though the company does not allow operators to use confidential files outside the company, it is impossible to reject any access requests to the files because both DAC and MAC determine whether to allow/disallow access requests based on the file's permissions (DAC) or security classification (MAC) regardless of the circumstances (location and/or time) under which the file is being accessed. Thus, in this scenario, conventional operating systems allow users to access confidential file if they have valid permissions (DAC) or a higher-level classification (MAC).

3 Adaptive Data Protection Mechanism

3.1 Overview of *Salvia*'s Data Protection Mechanism

The guidelines for the protection of privacy data [2] declared by the Organisation for Economic Co-operation and Development (OECD) or the ordinance based on OECD's guidelines oblige companies or persons who manipulate privacy data to use and manage it according to the intentions of the data owners. Thus, protecting privacy data does not imply enforcing the access restrictions defined by the data administrator, but thoes based on the policies configured by the data owner under the mutual agreement between the owner and the administrator. Therefore, it is necessary to apply individual access restrictions for each data entity. The goals of this research project are as follows:

- to realize a data protection mechanism suitable for protecting privacy data
- adapt this data protection mechanism to contexts
- make it compatible with the existing application programs

Note that in this study, we use the term "contexts" to refer to the parameters or hints for evaluating the threat of data leakage. Details regarding contexts are described in Sect. 3.2.

In the proposed approach, the operating system (*Salvia*) provides the data protection mechanism because it is capable of applying the mechanism transparently to existing application software without modifying or recompiling them. Thus, the proposed mechanism can ensure the compatibility of the existing applications. There are two file types in *Salvia*. One consists of regular files that are protected by conventional data protection mechanisms such as DAC. The other comprises of files that are protected by *Salvia*'s data protection mechanism. The difference between the former and latter is that the latter has certain related to data protection policies, but the former does not. Note that in this study, we use the term "privacy file" to refer to a file that has certain data protection policies associated with it.

To solve the problems described in Sect. 2, we propose a data protection mechanism with context-aware access control. The proposed mechanism can restrict data flows to within the proper scope of distribution. To achieve this functionality, we need to restrict the usage of computation resources that could serve as output channels for data leakage by any process accessing privacy files; these computation resources include files, pipes, sockets, and shared memory regions. In addition, if processes that have opened a privacy file (using the `open()` system call) invoke specific functions that could lead to data leakage, *Salvia* needs to determine whether to allow the process to invoke these functions or not. Therefore, according to the configuration of the data protection policies, *Salvia* needs to check and limit the execution of system calls that provide a service access to the resources that could lead to data leakage.

3.2 Contexts

In *Salvia*, the contexts are classified into "attribute contexts" and "environmental contexts" based on their acquisition method. Details regarding both attribute and environmental contexts are described in this subsection.

Attribute Contexts
Attribute contexts consist of the attributes and system call histories of a process. It is possible to acquire both of these from certain parameters managed by the kernel. The process attributes consist of user ID (UID), effective user ID (EUID), and process ID (PID). The system call histories consist of a system call number, its arguments and return value, and the time of which it was invoked. If the arguments contain pointers to user address space, *Salvia* copies the real arguments from the pointer-specified locations. The system call number is used by *Salvia* to determine whether a system call should or should not be controlled. These arguments are used to generate conditional expressions to control the system calls. The return value is used to record the result of execution of the system call or an identifier such such as a file descriptor.

Environmental Contexts
Environmental contexts comprise absolute time, relative time, and location information of users' computer. These parameters can be acquired from the device drivers of peripheral devices. The absolute time context represents the current time (real-world clock) and can be acquired from RTC. The relative time context represents the elapsed time from a particular absolute time that is regarded as time zero.

In *Salvia*, the radio wave intensities of wireless LAN; extended service set ID (ESSID), which is an identifier of the wireless LAN's access point; and the global positioning system (GPS) position data are available as environmental contexts representing location information. It is possible to acquire ESSID, radio wave intensity, and GPS position data from device drivers. If GPS position data is used as the context for location information, it is possible to obtain latitude and longitude data in order to express the location of a computer.

3.3 Classification of Controlled System Calls

From the viewpoint of preventing data leakage, the system calls based on the POSIX 1003.1 and controlled by *Salvia* are classified into three classes – **restrict class**, **context class**, and **manage class**. The system calls belonging to each of these classes are listed in Table 1.

restrict class: This class comprises system calls that could lead to data leakage (e.g., `read()`, `write()`, and `send()`). These system calls are controlled according to the data protection policies and contexts. Moreover, in order to use the results of the execution of these system calls as contexts, *Salvia* records the system call histories for this class.

context class: This class comprises system calls that could modify certain parameters related to context (e.g., EUID can be modified by `setuid()`). If these parameters are modified illegally, it could become impossible to enforce access restrictions on user processes. Therefore, *Salvia* records the system call histories for this class and prevents the parameters from being modified illegally.

manage class: This class comprises system calls that trigger the execution of pre-/post-processing procedures for managing system call histories. Since the system calls belonging to this class do not lead to data leakage, *Salvia* always permits these system calls. But in order to invoke pre-processing or post-processing functions, *Salvia* monitors these system calls.

Table 1. System call classification

Class	System calls (based on the POSIX 1003.1)
restrict	`read`, `write`, `readv`, `writev`, `pread`, `pwrite`, `mmap`, `munmap`, `shmat`, `send`, `sendmsg`, `sendto`, `execve`
context	`ioctl`, `setuid`, `setgid`, `setpgid`, `setreuid`, `setregid`, `settimeofday`, `stime`, `clock_settime`
manage	`open`, `close`, `socket`, `pipe`, `fork`, `vfork`, `_exit`, `kill`, `dup`, `dup2`, `mknod`, `fcntl`, `mq_open`, `mq_unlink`

In order to avoid data leakage, it is necessary to assign appropriate permissions to the system calls belonging to the **restrict class**. However, if one data protection policy is configured for each system call, the number of policies will increase. Subsequently, this approach will make it difficult to configure policies and/or will result in an increase in the probability of errors in the policy descriptions. To avoid these problems, we classify the system calls into four groups based on their procedure or scope of data propagation. These groups allow for the reduction in the number of data protection policies and ease the task of configuration. These groups are **read group**, **write group**, **send_local group**, and **send_remote group**. The system calls belonging to each of these groups are listed in Table 2.

read group: This group comprises system calls that read data from files.

write group: This group comprises system calls that write data to local files.

send_local group: This group comprises system calls that write data to other processes running on the local computer.

send_remote group: This group comprises system calls that write data to other processes running on remote computers.

Table 2. System call groups for the restrict class

Group	System calls
read	`read`, `readv`, `pread`, `mmap`, `munmap`
write	`write`, `writev`, `pwrite`, `mmap`, `munmap`
send_local	`write`, `writev`, `mmap`, `send`, `sendto`, `sendmsg`, `shmat`
send_remote	`write`, `writev`, `send`, `sendto`, `sendmsg`

3.4 Description of Data Protection Policies

The data protection policies consist of three parts – default policies, conditional expressions with contexts, and control directions for each system call group (Table 2). If no access control lists are specified in the data protection policies, all system calls are rejected fundamentally (according to the "default deny" principle); otherwise, the system calls may be permitted according to the data protection policies. Every time a system call is invoked, *Salvia* verifies whether the corresponding data protection policies are satisfied. These policies are written in XML. One of the advantages of using XML is that it ease the task of syntax verification by allowing the use of a program to automatically check the syntax.

Figure 1 shows the data type definition (DTD) for describing the data protection policies in *Salvia*; a brief example is shown in Fig. 2. In this figure, the **read**, **write**, **send_local**, and **send_remote** tags, which are the system call groups listed in Table 2, define the permissions for the execution of the corresponding system calls.

In particular, the **write** and the **send_remote** tags contain a few additional tags and attributes. The **write_access** tag defines the write access permission for files associated with the policies themselves; the **update** attribute also defines the write access permissions. The difference between the **write_access** tag and **update** attribute is that the permission defined by the **write_access** tag is enforced on a process after it has read the file, while the permission defined by the **update** attribute is enforced regardless of whether the process has read the file or not. As a result, all write accesses are always disallowed if both the **write_access** tag and **update** attribute are set to "deny," but the write accesses generated before the file is read are allowed if the **write_access** tag and the **update** attribute are set to "deny" and "allow," respectively. The **filename** tag defines the file or directory names to which write access is granted when the **write_access** tag is set to "deny." As a result, processes cannot write data to

```
<!ELEMENT data_protection_policy (default_access?,data_protection_domain+)>

  <!ELEMENT default_access (read?,write?,send_local?,send_remote?,syscall*)>
    <!ELEMENT read (#PCDATA)>
    <!ELEMENT write (write_access,filename*)>
      <!ELEMENT write_access (#PCDATA)>
      <!ATTLIST write_access update (deny|allow) "deny">
      <!ELEMENT filename (#PCDATA)>
    <!ELEMENT send_local (#PCDATA)>
    <!ELEMENT send_remote (send_remote_access,ip_address*)>
      <!ELEMENT send_remote_access (#PCDATA)>
      <!ELEMENT ip_address (#PCDATA)>
      <!ATTLIST ip_address version (4|6) "4">
    <!ELEMENT syscall (#PCDATA)>

  <!ELEMENT data_protection_domain (ACL)>
  <!ATTLIST data_protection_domain type (read|receive|both|none) "both">

    <!ELEMENT ACL (context,(access?|ACL*))>
      <!ELEMENT user (user_id+)>
        <!ELEMENT user_id (#PCDATA)>
        <!ATTLIST user_id type (effective|real) "real">
      <!ELEMENT group (group_id+)>
        <!ELEMENT group_id (#PCDATA)>
        <!ATTLIST group_id type (effective|own) "own">
      <!ELEMENT time (second*)>
        <!ELEMENT second (#PCDATA)>
        <!ATTLIST second mode (relative|absolute) "relative">
      <!ELEMENT location (device+)>
        <!ELEMENT device (value+)>
        <!ATTLIST device id CDATA "unknown">
          <!ELEMENT value (#PCDATA)>
          <!ATTLIST value type CDATA "integer">
      <!ELEMENT frequency (read?,write?)>
        <!ELEMENT read (#PCDATA)>
        <!ELEMENT write (#PCDATA)>

    <!ELEMENT access(read?,write?,send_local?,send_remote?,syscall*)>
      <!ELEMENT read (#PCDATA)>
      <!ELEMENT write (write_access,filename*)>
        <!ELEMENT write_access (#PCDATA)>
        <!ATTLIST write_access update (deny|allow) "deny">
      <!ELEMENT send_local (#PCDATA)>
      <!ELEMENT send_remote (send_remote_access,ip_address*)>
        <!ELEMENT send_remote_access (#PCDATA)>
        <!ELEMENT ip_address (#PCDATA)>
        <!ATTLIST ip_address version (4|6) "4">
      <!ELEMENT syscall (#PCDATA)>
```

Fig. 1. Document type definition (DTD) for *Salvia*'s data protection policies

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE data_protection_policy SYSTEM "policy.dtd">

<data_protection_policy>
  <default_access>
    <read>allow</read>
    <write>
      <write_access update="allow">deny</write_access>
    </write>
    <send_local>deny</send_local>
    <send_remote>
      <send_remote_access>deny</send_remote_access>
    </send_remote>
  </default_access>
  <data_protection_domain type="none">
    <ACL>
      <context>
        <gourp>
          <group_id type="own">1004<group_id>
        </group>
        <location>
          <device id="net_radio">
            <value type="character">officeroom</value>
            <value>50</value>
          </device>
        </location>
      </context>
      <access>
        <read>allow</read>
        <write>
          <write_access update="allow">deny</write_access>
        </write>
        <send_local>deny</send_local>
        <send_remote>
          <send_remote_access>deny</send_remote_access>
        </send_remote>
      </access>
    </ACL>
  </data_protection_domain>
</data_protection_policy>
```

Fig. 2. Brief example of data protection policies

any file except those specified by the **filename** tag. The **ip_address** tag, which is a subclass of the **send_remote** tag, specifies the destination hosts to which the file associated with the data protection policies can be sent.

The **default_access** tag defines the default action of processes and limits their permissions when none of the access control lists defined by the **ACL** tag are satisfied. The **data_protection_domain** tag defines the sets of access control lists and the **ACL** tag defines an access control list that consists of conditional expressions and control directions. The **ACL** tag entry can be defined to be greater than zero. An **ACL** tag entry of zero implies that only the default policies are defined in the data protection policies. The **context** tag defines conditional expressions with contexts. The **access** tag defines control directions for each system call group. In **ACL** and its subclass tags, nested tags have "AND relationships" and paralleled tags (declared at the same depth) have "OR relationships." If there are several policies that are satisified when a system call is invoked, *Salvia* chooses the first one according to the "first-match" principle.

The example data protection policy shown in Fig. 2 disallows all processes from making copies of associated files. However, the **update** attribute of the **write_access** tag is set to "allow," thus, the processes that do not read the file are allowed to write data to the file. This implies that access to the file is granted only those processes that intend to append and/or overwrite data.

Let us consider a case where a process reads a file associated with data protection policies and tries to send the file to another process. In this scenario, it is impossible to determine in the data protection policies whether the send request is legal or could cause data leakage. Thus, it becomes necessary to configure "fail-safe" policies, which are similar to the filtering rules of a packet firewall.

3.5 Enforcing Data Protection Policies with Contexts on Processes

All processes invoke the `open()` system call before they read and/or write a file. Therefore, in *Salvia*, `open()` is regarded as a trigger for enforcing access restrictions on these processes and *Salvia* reads the data protection policies as a pre-processing task for `open()`.

For example, if a process tries to read a file for which the data protection policies are configured, as shown in Fig. 2, *Salvia* refers to the system call histories in order to restrict system calls such as `write()` and `send()`. *Salvia* makes it possible to restrict write access to all files, pipes, and sockets as it refers to system call histories. Further, if additional access control lists with contexts are defined in the data protection policies, as shown in the latter part of Fig. 2 (e.g., if ESSID is *officeroom* and the radio wave intensity[1] is greater than or equal to *50*, then **read** and **write** are allowed but **send_local** and **send_remote** are disallowed), *Salvia* acquires the contexts (ESSID and radio wave intensity from device drivers) every time the system calls for the file are invoked and determines the permissions.

Salvia's data protection mechanism uses contexts in order to determine system call permissions. In particular, since *Salvia* refers a process's system call histories, the past operations of a process affect its current operations. Thus, a process that opens a file protected by data protection policies is restricted by *Salvia* until it is terminated.

[1] We regard the "link quality" status, which can be acquired via the device driver as shown in Fig. 5, as the radio wave intensity. Users may try to measure the link quality status before configuring the data protection policies if they use it.

3.6 Additional Examples

In this subsection, we discuss the effectiveness of *Salvia*'s data protection mechanism by providing practical data protection policies for the examples described in Sect. 2. In the first example, in which an operator unintentionally attaches a privacy file to an e-mail, the e-mail program process should not be permitted to use all sockets after it has read the privacy file. In order to achieve this in *Salvia*, both **send_local** and **send_remote** entries need to be set to "deny," as shown in Fig. 2. Even though the **write** entry is changed to "allow" to relax the restriction, *Salvia* still restricts the access to sockets. If the process tries to access a TCP socket via `write()`, the access request is rejected by *Salvia*'s kernel because (1) the kernel can detect which process invokes `write()`, (2) the kernel can also detect what kind of computation resources the process tries to access, and (3) the kernel rejects the execution of `write()` on TCP sockets according to the data protection policies.

In the second example, wherein a mobile computer in which confidential files are stored is carried illegally outside a company by the company's operator, *Salvia* does not permit any user processes to read the confidential files unless the contexts, especially the location of the mobile computer, satisfy the conditional expressions stated in the data protection policies, as shown in Fig. 2. User processes can read the confidential files only if the contexts satisfy both **group** and **location**. In this case, *Salvia* acquires the ESSID and radio wave intensity from the device driver whenever a process invokes `read()` and compares the acquired contexts (ESSID and radio wave intensity) with the conditional expressions (ESSID and radio wave intensity specified in the data protection policies). *Salvia* also compares the GID of the process with the configured GID. If both comparisons are successful, *Salvia* allows the process to access the confidential files according to the control directions in the data protection policies; in this case, the process is only allowed to read the confidential files.

4 Implementation

An early (prototype) version of *Salvia* has been implemented based on the Linux-2.6.6 kernel and now *Salvia* is being ported to the Linux-2.6.18 kernel. Figure 3 depicts the structure of *Salvia*'s data protection mechanism.

The memory pool provides memory allocation and object (a specialized data structure used in *Salvia*) caching functions for the other modules in *Salvia*. The context watcher acquires contexts by copying specific variables from the device drivers of peripheral devices such as wireless LAN or specific data streams from peripheral devices such as GPS and RFID.

The history logger is statically embedded in the kernel and the system call capture module, which is customized for each system call, is implemented by the loadable kernel module (LKM), as shown in Fig. 3. The history logger is invoked by the system call handler whenever system calls belonging to the **restrict**, **context**, or **manage** class (Table 1) are invoked by user processes. In *Salvia*, if a process controlled by *Salvia* (parent process) spawns a new (child) process,

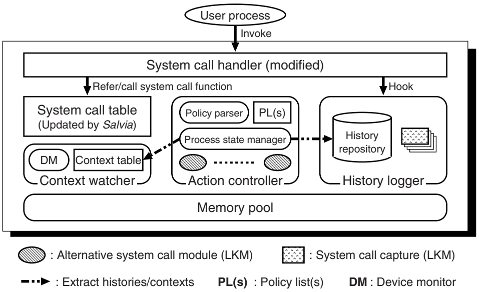


Fig. 3. Structure of *Salvia*'s data protection mechanism

the child process is also controlled by *Salvia* because the computation resources are shared by both the parent and the child processes. Therefore, *Salvia* makes the child process inherit the access restrictions and system call histories from the parent process in order to restrict the execution of the system calls. The history logger records histories as time-sequential data in the history repository. These histories are referred to by the action controller when it makes decisions regarding the permissions assigned to system calls. When a process is terminated, the process's histories are erased. Since the number and types of arguments differ for each system call, their acquisition functions are separated from the common functions of the history logger and are implemented for each system call by using LKM. However, in order to avoid bypassing or subverting *Salvia*'s access restrictions, *Salvia* needs to prevent user processes from disabling system call capture. Therefore, *Salvia* must restrict the invocation of `init_module()` and `delete_module()` because these system calls control the loading/unloading of LKMs.

The action controller is an embedded module that consists of the common functions of action controller, process state manager, and policy manager. The policy manager consists of the policy parser for the data protection policies and the policy list, which is a data structure for storing data protection policies parsed by the policy parser. Moreover, the alternative system call module is a subset module for restricting the system calls belonging to the **restrict** class. The alternative system call module is also implemented for each system call using LKM. The action controller controls file access operations (functions) invoked via system calls in order to realize file access restrictions. Therefore, the action controller overwrites the pointers of the access operations in a file object, which is generated by `open()`, with pointers of the alternative functions that determine file access permissions such as `read()` and `write()`. Moreover, pipe and socket access are also realized by pipe/socket access operations via a pipe/socket object generated by `pipe()` or `socket()`; thus, a method similar to the one used to enforce file access restrictions can be adapted to realize pipe/socket access restrictions.

Salvia needs to distinguish controlled processes (on which data protection policies are enforced) from other processes. In *Salvia*, we define several access control states in order to identify the processes that need to be controlled. These states are as follows: FREE, WATCHED, CONTROLLED, EXITING, and TERMINATED. The process state manager manages these access control states. All processes are forced to inherit the access control state of their parent process when they are spawned. This implies that a child process spawned by a parent process whose access control state is CONTROLLED inherits this state (CONTROLLED); further, this state is registered as the initial access control state of the spawned child process by the process state manager. The process state manager refer to the latest system call history to detect the changes in the access control state. If a process invokes `open()` with the write access permission, (for example, if the process state manager detects that a process is attempting to acquire a writable file that may serve as an output channel for data), the access control state of the process will be changed as follows: (1) if the latest state of the process is FREE, the state is changed into WATCHED, (2) if the latest state is WATCHED, it remains unchanged, or (3) if the latest state is CONTROLLED, it remains unchanged. In addition, the EXITING and TERMINATED states are used for managing system call histories. *Salvia* decreases the retain counters of each system call in the history for the EXITING state and deletes the history of the system call whose retain counter becomes zero for the TERMINATED state.

5 Evaluation

5.1 Overheads Involved in Logging System Call History

The proposal method for preventing data leakage may add certain overheads to the execution of each restricted system call. In this section, we evaluate the overheads involved in access restrictions in *Salvia*. The evaluation experiments were performed on an IBM compatible PC equipped with an Intel Pentium III 1.0 GHz processor, 256 MB RAM, and 40 GB Ultra ATA100 hard disk drive.

There may be two reasons for these overheads. One is the logging of system call histories and the other is determining system call permissions. In order to measure the overhead involved in logging system call histories, we prepared a microbenchmark that opens ten privacy files; this microbenchmark was executed on both *Salvia* (based on Linux-2.6.6) and on normal Linux-2.6.6. However, the disk access time hides the overhead of logging system call histories; this is because the files had been cached in the kernel's disk cache beforehand. In addition, we enabled the history logger but disabled the action controller in order to evaluate only the overhead of logging system call histories.

The results of the microbenchmark are shown in Fig. 4. It can be seen that the maximum difference between the processing times for the files (except file 1) in *Salvia* and normal Linux is 10.35 us. This implies that the additional 10.35 us is the overhead that consists of the times required for copying the arguments of `open()` and registering the system call history to the history repository. Figure 4 also shows that the processing time for file 1 in *Salvia* is 26.59 us more than normal Linux. This indicates that the time required to initialize the several data

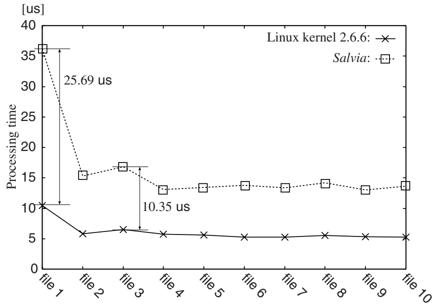


Fig. 4. Processing times of the `open()` system call

structures needed for managing system call histories is added to the overhead discussed above.

In addition, we conducted another experiment to measure the memory consumption of registering system call histories. According to one of the practical examples discussed in Sect. 2, we used Wanderlust 2.10.1, which is an e-mail client program running on GNU Emacs, to send an e-mail with a privacy file attached. We prepared three privacy files (sizes: 1 KB, 10 KB, and 100 KB) for this experiment.

The results of this experiment are shown in Table 3. These results indicate that the number of entries being registered in the system call histories increases with the size of the file being attached. In this experiment, about 2000 entries were registered in the system call histories and 62.80 KB of RAM was used.

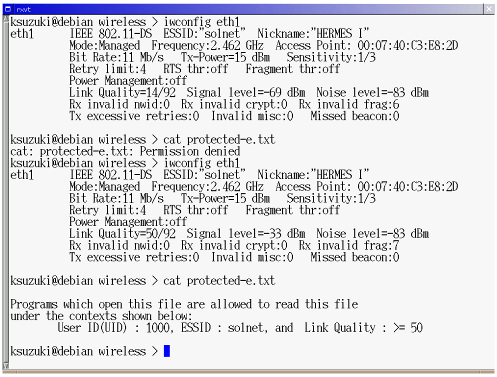
Table 3. Memory consumptions for registering system call histories

File Size	Open	Read		Write		Socket	Connect	Send	Recvfrom	Total	
		Sock	Other	Sock	Other					Count	Size
1 KB	29	10	62	46	41	5	5	4	4	206	7.42 KB
10 KB	30	9	33	208	36	5	5	4	4	334	11.49 KB
100 KB	31	8	47	1825	47	5	5	4	4	1976	62.80 KB

5.2 Adaptive System Call Limitation Based on Contexts and Data Protection Policies

We conducted an experiment to demonstrate the limitation of implementing system calls with context awareness. In this experiment, we used a popular command – `cat` – to display the contents of a privacy file. The data protection policies were configured as follows: (1) the data protection policies shown in Fig. 2 were used as the template policies, (2) all **default_access** entries were changed to "deny," and (3) both ESSID and GID were modified in the data protection policies – ESSID was changed to *solnet*, GID was changed to UID, and its value was changed to *1000*.

The results of this experiment are shown in Fig. 5. When the radio wave intensity (link quality) was *50*[2], *Salvia* allowed the `cat` command to invoke the `read()` operation. Therefore, the `cat` command successfully displayed the contents of the privacy file. However, when the radio wave intensity was *14*, *Salvia* disallowed the execution of the `read()`; therefore, the `cat` command failed to display the contents of the privacy file.



```
ksuzuki@debian wireless > iwconfig eth1
eth1      IEEE 802.11-DS  ESSID:"solnet"  Nickname:"HERMES I"
          Mode:Managed  Frequency:2.462 GHz  Access Point: 00:07:40:C3:E8:2D
          Bit Rate:11 Mb/s   Tx-Power=15 dBm   Sensitivity:1/3
          Retry limit:4   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=14/92  Signal level=-69 dBm  Noise level=-83 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:6
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

ksuzuki@debian wireless > cat protected-e.txt
cat: protected-e.txt: Permission denied
ksuzuki@debian wireless > iwconfig eth1
eth1      IEEE 802.11-DS  ESSID:"solnet"  Nickname:"HERMES I"
          Mode:Managed  Frequency:2.462 GHz  Access Point: 00:07:40:C3:E8:2D
          Bit Rate:11 Mb/s   Tx-Power=15 dBm   Sensitivity:1/3
          Retry limit:4   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=50/92  Signal level=-33 dBm  Noise level=-83 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:7
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

ksuzuki@debian wireless > cat protected-e.txt

Programs which open this file are allowed to read this file
under the contexts shown below:
        User ID(UID) : 1000, ESSID : solnet, and  Link Quality : >= 50

ksuzuki@debian wireless > 
```

Fig. 5. Demonstration of the limitation of implementing context-aware system calls

6 Related Work

The trusted platform module (TPM) [3] is a specification for hardware controller chips that provide safe storage spaces for storing information, safe work spaces for encryption/decryption, and ensure the integrity of information stored in the module itself. TPM is required by Microsoft's Next-Generation Secure Computing Base [4]. TPM can guarantee integrity of sensitive data such as the data protection policies. Moreover, *Salvia* can limit the scope of data propagation.

Information flow control is a software-based technique for data protection. The Bell-LaPadula Model [1] realizes information flow control and is one of the most popular model for MAC. In this model, the flow of information from the classified to the declassified level is not permitted but however, the Bell-LaPadula Model and other models for MAC impose access restrictions that are too strict to sustain the operation of user processes.

REMUS [5] is a kernel-level reference monitor for the prevention of the subverting execution of privileged programs such as daemon processes. In REMUS, all system calls are classified according to their level of threat with respect to system penetration; REMUS restricts the execution of system calls according to the policies configured for each system call. However, REMUS is not a

[2] The value (threshold) has been decided by our measurement. If the value is greater than or equal to 50, it indicates the computer that is used for the experiment is inside the room where the experiment has been made.

context-aware system; therefore, it is impossible to prevent data leakage in the situations described in Sect. 2.

Deeds [6], which is an extension of Java, provides a secure runtime environment for executing programs. Deeds records access histories of computation resources that need to be protected and imposes access restrictions on programs based on the histories. This approach is similar to that employed in *Salvia*; however, *Salvia* adapts its access control mechanism to not only the histories but also the attribute and environmental contexts described in Sect. 3.2. Moreover, Deeds has a limitation – it terminates a process if it dynamically loads a class during its exection. *Salvia* does not have this limitation.

Java information flow (Jif) is another extension of Java and is a privacy-aware programming language based on the decentralized label model [7]. The features of this model are as follows:

- assigning security labels to each data entity
- the label comprises two parts – data owner and reader – and controls read access
- this label can be modified by a user who is the owner of the data or who has the same authority as the owner
- if a variable is copied to another variable, the label of the copied variable is also copied

In Jif, it is possible to control information flows for protecting each variable; however, it is impossible to apply these control methods to existing applications without modifying of them. On the other hand, it is possible to apply *Salvia*'s access restriction mechanism without modifying existing applications.

7 Conclusion

In this study, we propose a data protection mechanism suitable for preventing leakage of privacy data. In particular, we describe the design, implementation, and evaluation of a privacy-aware operating system – *Salvia*. The features of *Salvia* are as follows:

- relating data protection policies to files containing privacy data
- protecting both protected file and data protection policies against illegal access
- limiting the execution of system calls adaptively based on data protection policies and context
- adopting system call histories as a form of context

Thus, *Salvia* is capable of preventing the leakage of privacy data regardless of users' mistakes or illegal operations. In addition, a child process spawned by a parent process whose access control state is either WATCHED or CONTROLLED is forced to inherit the system call histories of the parent process; thus, it is possible to restrict the execution of the system calls invoked by a child process according to the context it inherits from the parent process.

In Sect. 5.1, we show that the overheads of logging system call history and initializing data structures for registering them are 10.35 us and 25.69 us, respectively. Moreover, we also conduct several experiments to demonstrate the effectiveness of *Salvia*'s context-aware data protection mechanism. *Salvia* can restrict the execution of system calls according to the data protection policies and contexts, especially system call histories, relative/absolute time, and location.

References

1. Bell, D.E., LaPadula, L.J.: Secure Computer System: Unified Exposition and Multics Interpretation. MTR-2997, MITRE Corporation; ESD-TR-75-306 (May 1976)
2. Organisation for Economic Co-operation and Development: OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data (2004), http://www.oecd.org/document/18/0,2340,en_2649_201185_1815186_1_1_1_1,00.html
3. Trusted Computing Group: Trusted Platform Module (TPM) Specifications. https://www.trustedcomputinggroup.org/specs/TPM/
4. Microsoft Corporation: Next-Generation Secure Computing Base. http://www.microsoft.com/resources/ngscb/default.mspx
5. Bernaschi, M., Gabrielli, E., Mancini, L.V.: REMUS: A Security-enhanced Operating System. ACM Transactions on Information and System Security 5(1), 36–61 (2002)
6. Edjlali, G., Acharya, A., Chaudhary, V.: History-based Access Control for Mobile Code. In: Proceedings of the 5th ACM Conference on Computer and Communications Security, pp. 38–48 (1998)
7. Myers, A.C., Liskov, B.: Protecting Privacy using the Decentralized Label Model. ACM Transactions on Software Engineering and Methodology (TOSEM) 9(4), 410–442 (2000)

InfoCage: A Development and Evaluation of Confidential File Lifetime Monitoring Technology by Analyzing Events from File Systems and GUIs

Koji Kida[1,3], Hisashi Sakamoto[2], Hideo Shimazu[2], and Hiroyuki Tarumi[3]

[1] Service Platforms Research Laboratories
NEC Corporation
8916-47, Takayama-Cho, Ikoma, Nara, 630-0101, Japan
kida@da.jp.nec.com
[2] NEC System Technologies, Ltd., Japan
{sakamoto-hxb,shimazu-hxa}@necst.nec.co.jp
[3] Kagawa University, Japan
tarumi@eng.kagawa-u.ac.jp

Abstract. Information leakage from client PCs has grown into a serious problem in recent years. To address the problem, we propose file lifetime monitoring technology that enables users to accurately trace modifications of confidential documents by user commands such as copy, rename, and save, as well as copy/paste editing operations. The File Lifetime Monitoring Technology is based on analyzing primitive events from the file system and GUI and anticipates confidentiality risks using prepared knowledge of applications. Using monitoring results effectively, confidential information can be managed on workers'PCs. The prototype system has been successfully operated, and 243 co-workers tried it out. The trial shows that our system is practical in terms of performance degradation. Experiments show that both the quality and quantity of monitoring results are better than conventional monitoring software. In particular, the log size of our system can be reduced eleven times from the size of conventional software.

Keywords: Secure Office, Information Leakage Countemesure, Operation Monitoring, Log Analysis, Knowledge Base.

1 Introduction

This paper describes an information leakage countermeasure system based on our File Lifetime Monitoring Technology that enables users to accurately trace modifications of confidential documents by user commands such as copy, rename, and save, as well as copy/paste editing operations.

The motivation behind this research was two-fold. First, there seems to be no end to information leakage. A recent CSI/FBI computer crime and security survey said that information leakage continued to be the source of the greatest

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 246–261, 2007.
© Springer-Verlag Berlin Heidelberg 2007

financial losses [1]. In particular, leakage from end-user devices such as laptops and USB memories has emerged as a social issue in Japan because the number of this type of incident is increasing. So we decided to tackle the problem of leakage from end-user devices in order to meet such social needs.

Second, in enterprises, most confidential digital information is distributed as document files from word processors, spread sheet applications and so on. In fact, confidential document files have been leaked in many incidents. The simplest countermeasure might be to encrypt all confidential files. However, we don't think that will work well for the following reasons.

– Encrypted document files are inconvenient because users must decode them to their original files before use.
– It takes time to retrieve encrypted files because search engines cannot index them
– Key servers for encrypting documents are necessary, so their installation and operation costs become problematic.

We think it is important that end-user usability should not change before or after the installation of a countermeasure system. So we focus instead on monitoring the state of enormous numbers of confidential document files distributed in an enterprise.

2 Analysis of File Monitoring Method

2.1 Outline of Usual Method

We will begin by considering the usual method of client PC monitoring. There are some client PC monitoring products for grasping changes in files [2][3][4]. These systems capture a lot of events from an OS such as file open, mouse click, and application program launch, and output logs as monitoring results to the client PC's disk or monitoring server. The important point to note is that logs give only very primitive monitoring results. So the distinction between an illegal act and an unintentional act leading to an information leak cannot be made without detailed analysis. Conventional client monitoring products have log analysis tools and a security administrator can inspect and predict a user's behavior by using the log analysis tools. However, that is painstaking work. Therefore, it can be said that it's very difficult to detect a malicious act in real time. The products are used for their deterrent effect and for grasping the situation after an incident.

2.2 Problems of Usual Method

From the point of view of analyzing logs more easily and understanding file movements more exactly, the limits or problems of the usual client PC monitoring method are as follows.

File Name Change Problem. Usual methods monitor file movement based on the file name. So when a file name is changed, the log analyzer often cannot determine the relation between the original file and the file with a new name. Therefore, the movement of the file cannot be traced when a user changes the file name.

Data Reuse Problem. Almost all of the documents are editable and are modified by user copy buffer operations such as cut and paste commands among two or more files. So data flow between applications should be monitored in order to understand file movements exactly. However, monitoring is difficult because the file name isn't included in an event of copy buffer operation. Log analyzing administrators have to collect circumstantial evidence from another events such as an application use event, file system use event or GUI operation event.

Log Amount Problem. Original events from the OS are very primitive and all usual monitoring software outputs subsets of the events as monitoring results. Monitoring results are primitive, too. So an analysis of logs formulates a hypothesis and inspects logs from various angles. The more logs there are, the more the analytical accuracy improves. However, as the number of logs increases, the performance of a PC decreases. For this reason, the usual PC monitoring methods pose a dilemma.

3 Basic Idea of File Lifetime Monitoring

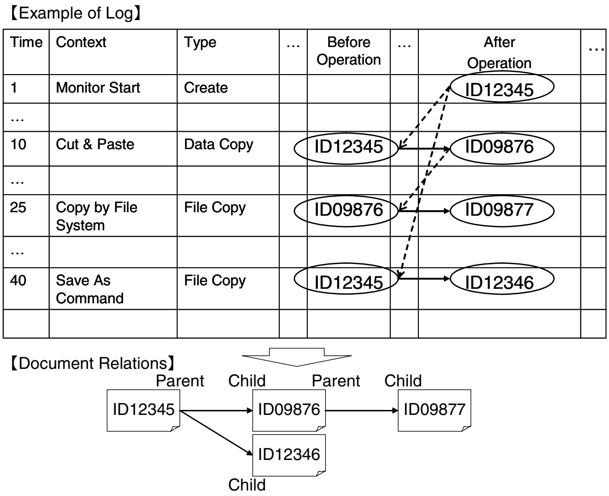
To address the problems, we propose file lifetime monitoring technology that enables users to accurately trace modifications of confidential documents by user commands such as copy, rename, and save, as well as copy/paste editing operations. In this section, we will begin by explaining what kind of log to output. We call the log an intelligent log. We also discuss whether the problems are solved by the intelligent log and explain how to make our intelligent log.

3.1 Intelligent Log

An example of our lifetime monitoring log is shown in Fig1. One record in the table in Fig.reffig:idea corresponds to one log instance. All log instances include the following five main fields:

- The **time** field represents the date-time of a user operation.
- The **context** field represents an advanced situation that our system presumed.
- The **type** field represents our category of file movement such as File Copy, Data Copy, Rename, and Attached Mail.
- The **before operation** hfield represents our file ID of the target file when beginning operation, e.g. the file ID of the source file when copying a file.
- The **after operation** field represents our file ID of the target file after operation, e.g. the file ID of the destination file when copying a file.

The important feature is that all records of the log have a before-operation field and an after-operation field. These fields are used for tracing file movement. We show the tracing of file movement using Fig.1. The first record shows that the document with ID12345 was created at time 1, and our system started monitoring it. At time 10, the record shows that the user cut from the ID12345 document and pasted to the ID09876 document. After that, the ID12345 and ID09876 documents were also copied by the ”Save As” command. Fig. 1 shows the parent-child relation. A source file is a parent, and a destination file is a child.



【Example of Log】

Time	Context	Type	…	Before Operation	…	After Operation	…
1	Monitor Start	Create				ID12345	
…							
10	Cut & Paste	Data Copy		ID12345		ID09876	
…							
25	Copy by File System	File Copy		ID09876		ID09877	
…							
40	Save As Command	File Copy		ID12345		ID12346	

【Document Relations】

Fig. 1. Example of File Lifetime Log. The important feature is that all records of the log have a before-operation field and an after-operation field. These fields are used for tracing file movement.

3.2 Solving Problems by the Intelligent Log

File Name Change Problem. The intelligent log does not use a file’s name. All target files are assigned our file ID based on UUID (Universally Unique Identifier). In case of a change in a file’s name, the intelligent log shows that the old file name is the parent and the new file name is its child. When a file name changes, we can trace its movement using the parent-child relation. Thus, the file name-change problem is solved.

Data Reuse Problem. The intelligent log includes the target file name of a user operation. Even when using a copy buffer on some applications, the log includes the source file name and the destination file name. The intelligent log is created by analyzing primitive events from the file system and GUI in real time and anticipates the context by matching knowledge of the applications. So the intelligent log is not primitive but advanced, and the data reuse problem is solved.

Log Amount Proble. The intelligent log is not primitive but advanced as stated. In the case of primitive logs, the analysis of logs formulates a hypothesis and inspects logs from various angles. So a lot of logs are necessary to improve analytical accuracy. On the other hand, the advanced log analysis succeeds even if the amount of logs is small. Trials with 243 co-workers showed that the log size of our system can be reduced eleven times from the size of conventional software.

4 File Lifetime Monitoring Technology

4.1 Comparison of the Processing Flow

In this section, to explain the features of our File Lifetime Monitoring Technology, we first explain the differences between the macro processing flows of conventional monitoring methods and those of our monitoring method. Conventional systems operate with the following flow.

1. A lot of primitive events from the OS are captured.
2. A necessary event is selected by a user's filtering configurations.
3. Selected events are output as monitoring logs.
4. Huge numbers of logs are managed by security administrators.
5. A log is analyzed if necessary.

On the other hand, our system works with the following flow.

1. A lot of primitive events from the OS are captured.
2. Captured events are analyzed by defining user behaviors in the knowledge base in real time.
3. Analyzed results are output as monitoring logs.
4. Huge numbers of logs are managed by security
5. A log is analyzed if necessary.

The most important point is the difference in the processing of (2). Our method analyzes primitive events and predicts the context in real time while the conventional method only chooses primitive events. So the quality of the logs processed in (3) is quite different. Our logs are advanced while conventional logs are primitive. Though both methods are basically the same in (4) and (5), the conventional method is more time-consuming than our method. This is because the amount and the quality of the logs are significantly different. After all, it is a key point how high the quality of analysis is in the processing of (2). We have developed File Lifetime Monitoring Technology to achieve this processing of (2).

4.2 Architecture of File Lifetime Monitoring Technology

The architecture of our system is shown in Fig. 2.

The File Lifetime Monitoring Technology is a technology that presumes the semantics from a user's operation events and events from the file system. A user's operation events are primitive GUI operation events such as a menu click

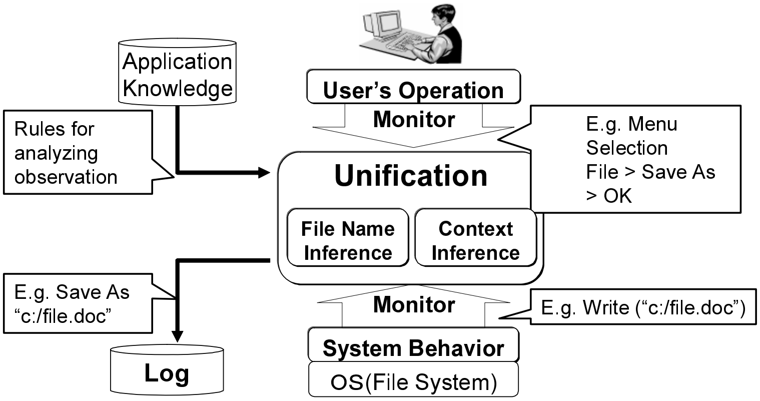


Fig. 2. Architecture of our File Lifetime Monitoring Technology. The technology presumes the semantics from a user's operation events and events from the file system.

with a mouse device or input from a keyboard. Events from the file system are primitive file system API calls from applications such as open(), read(), write(), and close(). A conventional system outputs these events directly. n the other hand, our system infers semantics.

The semantic inference consists of a context inference and a file name inference. Context inference means what the user is doing with a client PC at the present time and file name inference means what the file names are of the documents a user is using on a client PC at the present time.

The semantic inference is performed by unifying a user's operation events, events from the file system and application knowledge.

Application knowledge is a set of defined application behaviors. Application knowledge is described by defining the semantics for the sequences of the user operation event and the sequences of the file operation event. For example, application knowledge assigns the "Save As" function for a series of following events as:

1. User clicks "File" menu item
2. User clicks "Save As ..." submenu item
3. Application opens dialog of file name input
4. User inputs file name by keyboard
5. User presses OK button by mouse
6. Application calls Open() API
7. Application calls Write() API
8. Application calls Close() API.

Items (1)–(5) are concerned with user operation, and items (6)–(8) are concerned with file operation. In this example, unification means checking consistency in the user input file name in (4) and the file name of the file access in (6)–(8).

Finally, our system outputs an intelligent log that includes the client PC context, source file name and destination file name. This information shows the parent-child relation of the files, so a security administrator can trace file movement.

4.3 File Name Inference

The logs based on File Lifetime Monitoring Technology include the source file name, destination file name, operation, use application, user information, PC information and so on. In particular, the file name and correlation between documents can be monitored exactly. So analyzing the logs enables a user to trace a document to its source. In this section, we propose two technologies to infer the file name.

File Access Consistency Check Method. This is mainly used in order to infer the destination file name. The method of extracting file names is based on the side effects of unifying a user's operation events, file access events and the application knowledge. The file names in the application knowledge are described as variable. After unifying them successfully, all values of the variables that include the source file name are determined. In the above example, the file name in (4) and the file name of file access in (6)–(8) correspond with the same variable of the application knowledge. In this case, the variable means the destination file. So the value of the variable that is the result of unification is used as the destination file name in the log.

Active File Extraction Method. This is mainly used in order to infer the source file name. Though a user can use plural files on plural applications at the same time through the multitask function of an OS, the file in the top window is unique. The top window file, called the active file, corresponds with the source file in almost all contexts. This method infers the file name based on the active file history that is information on the chronological changes in the active file in every application. The active file history is built by paying attention to the change in the window title at each read file, and relating the file to the window title.

5 InfoCage: Confidential File Movement Trace Application

5.1 Outline of InfoCage

We have developed a client PC security application based on our File Lifetime Monitoring Technology for feasibility assessment. We call the application InfoCage. InfoCage allows a user to trace movements of many confidential files in a client PC.InfoCage is a client-server type system. In InfoCage, client means a worker's PC in which a monitoring agent resides. A server means a log management server for an organization or a task group (see Fig. 4).

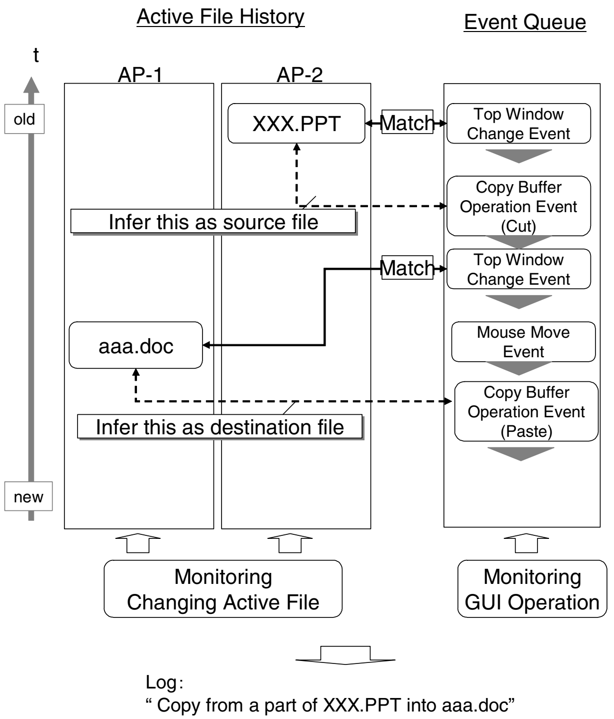

Active File History
Event Queue
t
old
new
AP-1
AP-2
XXX.PPT
Match
Top Window Change Event
Copy Buffer Operation Event (Cut)
Infer this as source file
Match
Top Window Change Event
Mouse Move Event
aaa.doc
Copy Buffer Operation Event (Paste)
Infer this as destination file
Monitoring Changing Active File
Monitoring GUI Operation
Log:
" Copy from a part of XXX.PPT into aaa.doc"

Fig. 3. File name inference by active file monitoring

–CLIENT–

Target File Configuration Tools. InfoCage does not monitor all files but only user selected files. This tool enables a user to define monitored files in a variety of ways. For example, a user can define documents that are carried from a certain server to a client PC as monitoring targets.

Monitoring Agent. This is resident software in a client PC for monitoring. This is based on our File Lifetime Monitoring Technology. This uploads monitoring results to the Log Management Server through a network. In order to defend against a malicious user, the log is encrypted and no user can kill the process of the agent.

Application Knowledge. Application knowledge is not built by users but is built and distributed by security administrators. In this trial, we investigated applications to use with subjects beforehand, and we built application knowledge and distributed it. The application knowledge includes the most typical applications such as Microsoft Office, Microsoft Internet Explorer, and Firefox.

–SERVER–

Log Management Server. This is a database that collects the log uploaded from the monitoring agents. It is usually implemented in a conventional relational database.

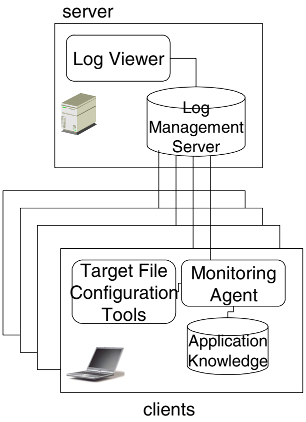


Fig. 4. InfoCage is a client-server type system. In InfoCage, client means a worker's PC in which a monitoring agent resides. A server means a log management server for an organization or a task group.

Log Viewer. This is a viewer that refers to the logs accumulated in the log management server. This viewer indicates the parent-child relationships of monitored files, a list of the confidential files in the client PC and a list of the files copied to USB memory or a network drive (see Fig.6) Usage of the tool is explained in the next section.

5.2 Examples of Use

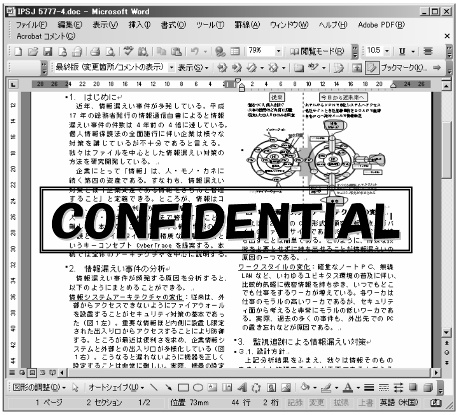
In case of daily office job of worker. A user can work using a PC as in the past except for the following two points.

– At the time of making a confidential document: A user should define it as a monitored document using the Target File Configuration Tools just after the user has made a confidential file.
– At the time of using a confidential document: The usage is basically no different from ordinary usage, except that when opening a confidential file, a user is warned that the file will be monitored as shown in Fig.5.

In case of log analysis of security administrator. When an administrator can specify the file that he/she wants to examine to some extent, he/she inputs conditions such as a file name, a file extension, a kind of operation, the date and time and searches for it. He/She selects a file from a list of files in the search results and can see its movement. He/She can also observe the parenthood of this file and can follow the log of a corresponding file in succession (see Fig.6).

5.3 System Implementations

The InfoCage prototype system has been successfully operated on Microsoft Windows XP and 2000. Capturing GUI operation is implemented by MSAA


IPSJ 5777-4.doc – Microsoft Word
ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ツール(T) 罫線(A) ウィンドウ(W) ヘルプ(H) Adobe PDF(B)
Acrobat コメント(C)
79%
閲覧モード(R)
10.5
最終版（変更箇所/コメントの表示）
表示(S)
ブックマーク(K)...
図形の調整(D)
オートシェイプ(U)
1 ページ 2 セクション 1/2 位置 73mm 44 行 2 桁 記録 変更 拡張 上書 英語（米国）
従来
今日から近未来へ

CONFIDENTIAL

1. はじめに

近年、情報漏えい事件が多発している。平成 17 年の総務省発行の情報通信白書によると情報漏えい事件の件数は 4 年前の 4 倍に達している。個人情報保護法の全面施行に伴い企業は様々な対策を講じているが不十分であると言える。我々はファイルを中心とした情報漏えい対策の方法を研究開発している。

企業にとって「情報」は、人・モノ・カネに続く第四の資産である。すなわち、情報漏えい対策とは「企業資産である情報をきちんと管理すること」と定義できる。ところが、情報はコ[illegible]いうキーコンセプト CyberTrace を提案する。本稿では全体のアーキテクチャを中心に説明する。

2. 情報漏えい事件の分析

情報漏えい事件が頻発する原因を分析すると、以下のようにまとめることができる。

情報システムアーキテクチャの変化：従来は、外部からアクセスできないようにファイアウォールを設置することがセキュリティ対策の基本であった（図 1 左）。重要な情報ほど内側に設置し限定された出入り口からアクセスすることにより防御する。ところが最近は便利さを求め、企業情報システムと外部との出入り口が多様化している（図 1 右）。こうなると漏れないように機器を正しく設定することは非常に難しい。実際、機器の設定

[illegible]ち出すことは簡単である。このように、特殊な技術を必要とせずに持ち出せることが情報漏えいの原因の一つである。

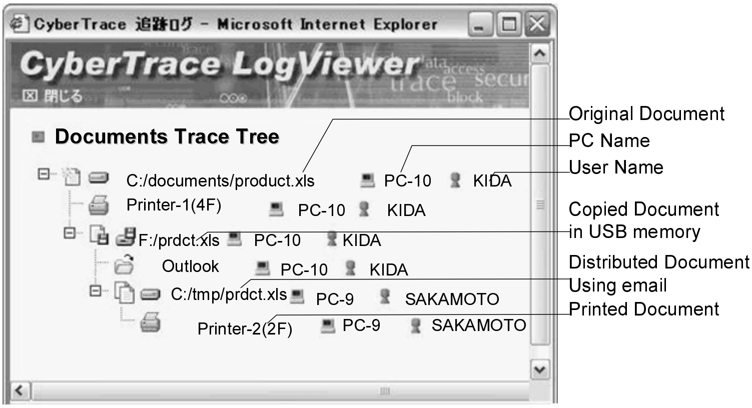
ワークスタイルの変化：軽量なノートＰＣ、無線 LAN など、いわゆるユビキタス環境の普及に伴い、比較的気軽に機密情報を持ち歩き、いつでもどこでも仕事をするワーカが増えている。各ワーカは仕事のモラルの高いワーカであるが、セキュリティ面から考えると非常にモラルの低いワーカである。実際、過去の多くの事件も、外出先での PC の置き忘れなどが原因である。

3. 監視追跡による情報漏えい対策

3.1. 設計方針

上記分析結果をふまえ、我々は情報そのもの

Fig. 5. Screen shot when launching confidential document



Documents Trace Tree

C:/documents/product.xls PC-10 KIDA
Printer-1(4F) PC-10 KIDA
F:/prdct.xls PC-10 KIDA
Outlook PC-10 KIDA
C:/tmp/prdct.xls PC-9 SAKAMOTO
Printer-2(2F) PC-9 SAKAMOTO

Fig. 6. Example of Log Viewer screen shot. This viewer indicates parent-child relationships of monitored files.

(Microsoft Active Accessibility) API[5], which provides a standard mechanism in order to exchange GUI operation information. Capturing file access is implemented by an API hook technique that injects a user library into the application process and captures the Win32 API calls of the applications [6]. Our File Lifetime Monitoring Technology and Log Viewer are implemented by Microsoft Visual C++. The data format of the application knowledge and log is XML. The log management server is implemented by the Microsoft SQL Server 2000 Desktop Engine (MSDN2000).

What one should note here is that InfoCage isn't dependent on the kind of platform, e.g. OS, computer language, and database, for implementations. For example, if the filter driver is used, the capturing of file access and GUI operation is possible for UNIX.

6 Evaluations

6.1 Method

Subjects. Two hundred forty-three workers in our research section and a business development department tested the system for 3 months.

(Experiment-1) Evaluation for Quantity of Logs. The subjects performed certain work both on a PC installed with InfoCage and on a PC installed with a conventional monitoring system. We compared the quantity of the logs from these systems. The work consisted of 14 kinds of operations related to information leakage such as file copying to USB memory, data recycling using a copy buffer, and document name change using the "Save As" menu item. InfoTrace (Version 1.4) [2] was adopted as the conventional monitoring system for the following reasons.

- InfoTrace is single function software for file monitoring in a client PC and is comparatively similar to the purpose of our system.
- InfoTrace is used widely in the Japanese market, and it seems to perform well.
- InfoTrace is customizable for monitoring certain items.

However, we have selected six subjects at random among the 243 workers in the interest of saving time.

(Experiment-2) Evaluation for Quality of Logs. We analyzed the logs from Experiment-2 and evaluated them from the point of view of how many of the following kinds of files could be detected:

- List of confidential documents open to the public to network drive
- List of confidential documents in the client PC
- List of confidential documents that should be deleted and have been deleted.

(Experiment-3) Evaluation for Decline of Performance. We compared performances of a PC installed with InfoCage and a PC installed with InfoTrace using a performance benchmark tool. We selected two portable PCs and two desktop PCs. We also asked all subjects the following two questions:
"Did you feel a decrease in performance?"
"If you did feel a decrease in performance, when did you feel so?"

6.2 Results

(Experiment-1) Evaluation for Quantity of Logs. As for InfoTrace, the amount of logs changes according to the filtering setting of the monitoring items. To make conditions fair between InfoCage and InfoTrace, the following were set as the monitor items:

- File access by filers such as Microsoft Explorer
- File access by applications such as Microsoft Office
- Copy buffer access.

The logs of InfoTrace include fields where it seems not to use the domain name, the mac-address and the logon user name for this analysis. So we evaluated the number of records as the amount of logs. The result is shown in Fig.7. The log amount of InfoCage can be reduced eleven times from the amount of InfoTrace. Therefore, we can reasonably conclude that the quantity of monitoring results of InfoCage is better than that of InfoTrace.

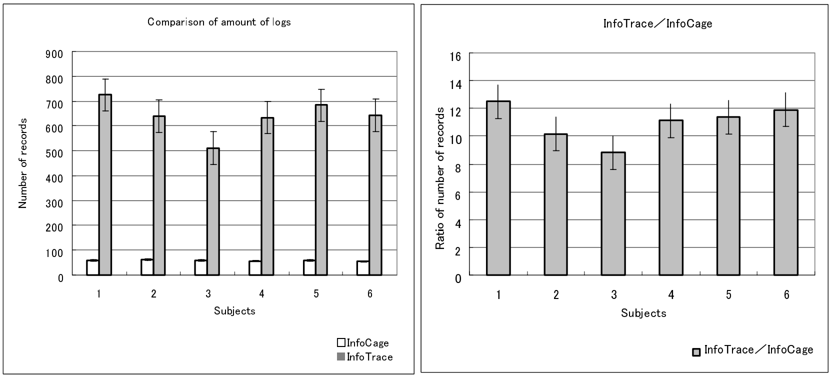
(Experiment-2) Evaluation for Quality of Logs. The result for the quality of logs is shown in Fig.8. We think a factor in this gap is that InfoTrace cannot trace, but InfoCage can trace, file movement in the following operations.

- Operations of document name change using application, i.e. the Save As command
- Operations of data copy using copy buffer between certain documents, i.e. the Cut and Paste commands

InfoCage could detect all of the file movements except the following case.

User Operation 1. Open new file. (We call it NewDoc.)
2. Open a monitored file. (We call it ConfidentialDoc.)
3. Open a non-monitored file. (We call it NormalDoc.)
4. A part of the data on ConfidentialDoc is copied to NewDoc.
5. A part of the data on NewDoc is copied to NormalDoc.

In this case, NormalDoc is a related document of ConfidentialDoc because NewDoc is a child document of ConfidentialDoc, and NormalDoc is a child document of NewDoc. However, InfoCage cannot trace such parent-child relations



	Subjects	1	2	3	4	5	6	**average**	**standard deviation**
Number of records	InfoCage	58	63	58	57	60	54	**58.3**	**2.7**
	InfoTrace	725	640	511	634	683	643	**639.3**	**65.5**
	InfoTrace／InfoCage	13	10	8.8	11	11	12	**10.96**	**1.2**

Fig. 7. Result of amount of logs. The log amount of InfoCage can be reduced eleven times from the amount of InfoTrace.

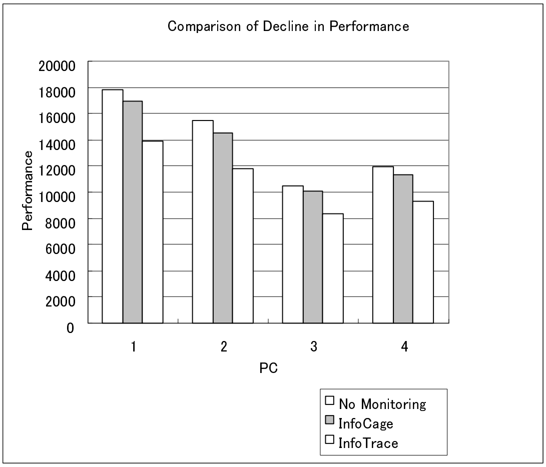
Evaluation Item	Answer	InfoCage		InfoTrace	
(A)	5	5	100.0%	2	40.0%
(B)	45	44	97.8%	14	31.1%
(C)	10	10	100.0%	1	10.0%

(A) List of confidential documents open to public to network drive

(B) List of confidential documents in client PC

(C) List of confidential documents that should be deleted and have been deleted

Fig. 8. The result of the quality of the log. A factor in this gap is that InfoTrace cannot trace, but InfoCage can trace, file movement in operations of document name change using application, and operations of data copy using copy buffer between certain documents.



	PC	1	2	3	4	**Average**	
Performance	No Monitoring	17848	15494	10489	11938	**13942.3**	
	InfoCage	16956	14564	10069	11341	**13232.6**	**94.9%**
	InfoTrace	13904	11797	8358	9300	**10839.5**	**81.9%**

Fig. 9. Results of evaluation for decline in performance. We think a factor in this gap is that InfoTrace cannot trace, but InfoCage can trace, file movement in certain operations.

because NewDoc doesn't have file name yet. We think this issue can be improved by assigning NewDoc a dummy file until a user saves it with a new file name explicitly. We think that InfoCage can solve this problem by remodeling the application knowledge. Therefore, we can reasonably conclude that our InfoCage is not a perfect monitoring system, but its architecture is improvable by remodeling the application knowledge and the quality of its monitoring results is better than that of InfoTrace.

(Experiment-3) Evaluation for Decline of Performance. To make the conditions fair between InfoCage and InfoTrace, we experimented only with settings that did not use any function that either InfoCage or InfoTrace doesn't have. For example, we didn't use the encryption function of logs because InfoTrace doesn't have that function. The result of PC performance is shown in Fig.9. Unfortunately, the measure of performance is a peculiar measure of the benchmark software, and we are not sure how to estimate performance. But this standard means that performance is high so the value is big. For InfoCage, it was a decline in performance of around 5% with all PCs. On the other hand, it was a decline of around 20% for InfoTrace. Results of questionnaires show that InfoCage is faster than InfoTrace. Some subjects said that they often felt stressed because of a decline in the performance of InfoTrace. No subject felt stressed by the decline in performance of InfoCage. Therefore, we can reasonably conclude that InfoCage is practical in terms of performance degradation.

7 Discussions

7.1 Application Knowledge

The most time-consuming part in the introduction of InfoCage is to make the application knowledge. We have to describe the knowledge of all behaviors of all applications that should be monitored. The numbers of the rules of application knowledge for the trial are as follows:

- The Number of Registered Applications: **59**
- The Number of Registered Contexts: **658**

The programmer should build these rules with a special tool [7]. We think that it's simple work for those who know how to use the tool, but it's a troublesome task. In addition, the method of testing the application knowledge is a problem. The application knowledge should be evaluated from the following viewpoints.

- Soundness: This means that no rule includes mistakes.
- Completeness: This means that rules cover all behaviors of all applications.

This time, we used the same technique as the test for software development. This method can check for soundness. However the check for completeness is difficult. The methods for building the application knowledge efficiently and for checking its completeness remain as challenges for the future.

7.2 Cooperation with Other Countermeasure Systems

The purpose of this research is to make a client PC secure. The approach of monitoring is one among several countermeasures. For example, other approaches are shown below.

Encryption. Countermeasure for theft of portable PC by coding hard disk and removable disk entirely.

Antivirus. Countermeasure for information leakage by virus.

Limiting Operation. Countermeasure by prohibiting application operation when possibility to leak information exists.

Configuration Check. Countermeasure by determining mistakes in security settings of PCs.

No countermeasure is perfect. It is most important to coordinate these countermeasures. In the following, we give examples of coordination between InfoCage and other countermeasures.

Coordination between InfoCage and Encryption Tools. Encrypted document files are inconvenient because a user must decode them to their original files before using them. In other words, usability is a problem for encryption tools. InfoCage can manage important files and related files of important files, so it can distinguish important files from non-important files. When coordinating InfoCage with encryption tools, only important sentences can be encrypted. The coordination system can improve end-user usability.

Coordination between InfoCage and Configuration Check Tools. We'll consider the example of checking whether the following security policy is defended.

> Security Policy: Important sentences prohibit putting a file in the network shared folder.

A conventional configuration check tool can check the document only superficially. T he tool cannot check based on the importance of a document, so it cannot check the above security policy. When coordinating InfoCage with a configuration check tool, it can check for an inconsistent point in the importance of a file and in equipment settings. So the coordinated system can check the above security policy.

8 Conclusion

Information leakage from client PCs has grown into a serious problem in recent years. To address the problem, we have proposed File Lifetime Monitoring Technology that enables users to accurately trace modifications of confidential documents by user commands such as copy, rename, and save, as well as copy/paste editing operations. The File Lifetime Monitoring Technology is based on analyzing primitive events from the file system and GUI and anticipates confidential risks using prepared knowledge of applications. Using monitoring results effectively, confidential information can be managed on workers'PCs. The prototype system has been successfully operated, and 243 co-workers tried out our system. The trial shows that our system is practical in terms of performance degradation. Experiments show that both the quality and quantity of monitoring results is better than conventional monitoring software. In particular, the log size of our system can be reduced eleven times from the size of conventional software.

References

1. CSI: 2003 CSI/FBI Computer Crime and Security Survey. 8th annual (2003)
2. Soliton Systems K.K Home Page: InfoTrace Product Information (in Japanese)., http://www.soliton.co.jp/products/infotrace/
3. IRI Commerce and Technology Inc: AllWatch Product Information (in Japanese). http://www.iri-ct.co.jp/allwatcher/
4. MOTEX K.K Home Page: LanScopeCAT5 Product Information (in Japanese). http://www.motex.co.jp/cat5/top.shtml
5. Microsoft Home Page: Microsoft Active Accessibility. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msaa/msaastart_9w2t.asp
6. J. Richter: Microsoft Press, 3rd edn. (1997)
7. Microsoft Home Page: FileMon for Windows v7.04. http://www.microsoft.com/technet/sysinternals/utilities/filemon.mspx

Accredited Symmetrically Private Information Retrieval

Mohamed Layouni

School of Computer Science, McGill University,
3480 University Street, Montreal, H3A 2A7, Quebec, Canada

Abstract. With the digitization of society and the continuous migration of services to the electronic world, individuals have lost significant control over their data. In this paper, we consider the problem of protecting personal information subjects. More specifically, we propose a new primitive allowing a data subject to decide when, how, and by whom his data can be accessed, without the database manager learning anything about his identity, at the time the data is retrieved. The proposed solution, which we call *Accredited SPIR*, combines symmetrically private information retrieval and privacy-preserving digital credentials. We present three constructions based on the discrete logarithm and RSA problems. Despite the added privacy safeguards, the extra cost incurred by our constructions is negligeable compared to that of the underlying building blocks.

Keywords: Symmetrically private information retrieval, anonymous credentials, policy enforcement.

1 Introduction

In a transaction-based world, with continuously shrinking resources, access control has always been, and still continue to be a central issue. Oftentimes, to benefit from a service or a resource, one is asked to show his identity, or prove possession of a set of qualifications and privileges. In many cases, this forces individuals into leaving identity trails behind them, which could be used for criminal activities such as unlawful monitoring and identity theft. The data collected from such interactions, although generally rich in personal information, is in most cases stored in databases lying outside the control of the data subject. Various techniques have been proposed in the past to strengthen users' privacy and help protect their personal information. Among these we note privacy preserving digital credentials [Cha85, Bra00, CL02], and symmetrically private information retrieval protocols [GIKM98, CMO00, KO97, AIR01, Lip05].

In a symmetrically private information retrieval (SPIR) system, there are generally two players: a Sender and a Receiver. The Sender has a database DB of records, and the Receiver submits a query Q to the Sender in order to retrieve a particular record. The main requirement in a SPIR system is privacy for both the Sender and the Receiver. That is, on the one hand the Sender should not

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 262–277, 2007.
© Springer-Verlag Berlin Heidelberg 2007

learn any information about the index of the record the Receiver is interested in, and on the other hand, the Receiver should not learn any information about the database, beyond the content of the record defined in the query Q, and what is already publicly known. In particular, the Receiver should not be able to learn information about more than one record per query. For instance, the Receiver should not be able to learn, through one query, the value of any function on a set of more than one record. SPIR systems have many real-life applications; for instance, consider a scenario where the inventor of a new drug needs information on a number of chemical components that will constitute his final product. This information can be accessed *for a fee* at some central database. This database could be managed, however, by parties with possibly competitive interests, and the inventor fears that his intellectual property (IP) will be compromised. He would like, therefore, that his queries remain concealed from the database manager. The latter, on the other hand, wants to be paid for all information retrieved from his database. It is clear that the SPIR system described above, can be a solution to this set of conflicting requirements.

There are similar applications however, that are closely related to the IP example above, which cannot be solved by a SPIR primitive. Consider for example the following e-health scenario where three types of participants are involved: (1) a patient, (2) a medical database containing the health records of patients, and (3) a doctor querying the medical database on patients' health records. The medical database and the doctor can be thought of as the Sender and Receiver, respectively, in a traditional SPIR setting. The requirements in the e-health application are as follows:

1. **Privacy for the Receiver:** The Receiver (doctor) wants to retrieve records from the medical database, without the Sender (DB) learning the index of those records, and thus the identity of his patient.
2. **Privacy for the Sender:** The Sender (DB) wants to be sure that, for each query, the Receiver (doctor) learns information only on one record (defined in the query) and nothing about the other records.
3. **Privacy for the data subject:** In order to comply with privacy legislation, the Sender wants to be sure that the Receiver has an valid reading authorization from the owner of the targeted record (i.e., the patient). We call the latter, *an Authorizer*. Notice that the Sender should not be able to learn the Authorizer's identity, otherwise the first requirement will be violated.

The example above shows a typical scenario where plain SPIR primitives fall short of protecting the interests of the Sender, the Receiver, and the Authorizer at the same time. The solution we provide in this paper, addresses the interests of all three parties, and solves the problems described above. We call the presented solution: *Accredited SPIR*. In what follows, we sometimes refer to the latter set of requirements, namely privacy for the data-subject, the Sender, and the Receiver, as the *Accredited SPIR problem*.

Solution highlight. In the Accredited SPIR setting we have three players: a Sender, a Receiver, and an Authorizer. The Receiver submits a query Q to the Sender, who replies with a response R. The Receiver recovers the answer to his query from R. The main contribution of Accredited SPIR, is to assure the Sender, before processing the query Q, that the Receiver has obtained an explicit consent from the owner of the record defined in Q, *without* revealing the identity of this owner (i.e., the Authorizer).

The Accredited SPIR architecture we propose, combines three cryptographic primitives: privacy-preserving digital credentials, homomorphic encryption, and SPIR systems. Privacy-preserving digital credentials [Cha85, CP92, Bra00, CL02, CL04] are cryptographic tokens issued by a certification authority CA to individuals. The CA encodes in each credential a set of attributes about the identity of its recipient. The latter is called a credential holder. A credential holder may later show his credential to a verifier in return for a service or a privilege (e.g., to receive medical treatment). Unlike traditional PKI certificates (e.g., X.509), privacy-preserving digital credentials allow their holders to selectively disclose information about their attributes [Bra00]. In particular, if a credential holder has a set of attributes $(x_1, \cdots, x_n)$, then he can prove any predicate $\mathcal{P}$ satisfied by those attributes, without the verifier learning any extra information beyond the status of $\mathcal{P}(x_1, \cdots, x_n)$.

Assume the Authorizer has a CA-issued identity credential *Cred* containing a set of attributes (ID,Age,$\cdots$). The idea is to first make the Authorizer and Receiver jointly compute the query Q, and then have the Authorizer produce a signed proof of knowledge of the secret attributes embedded in *Cred*. Along with the latter, the Authorizer proves that the ID attribute embedded in *Cred* is the same as the one contained in the query Q. The Receiver then deposits the signed proof along with the query to the Sender. The Sender first checks the validity of the proof. If accepted it carries on with the SPIR protocol and processes the query, otherwise it rejects.

As mentioned earlier, the signed proof does not reveal any information about the credential holder, and yet guarantees that the content of the query is consistent with the secret identity attribute embedded in the credential. Furthermore, owing to the fact that it is hard for a polytime adversary to forge credentials, or make proofs about credentials he does not own, the Sender can be sure that the Receiver has indeed obtained an explicit consent from the targeted record's owner.

This paper presents three constructions to solve the accredited SPIR problem. The first is based on a modified version of one of Brands DL-based credentials [Bra00, Section 4.5.2], the ElGamal cryptosystem, and a SPIR system proposed by Lipmaa in [Lip05]. The two additional constructions are variants of the first, and use an RSA-based version of Brands credentials [Bra00, Section 4.2.2], in combination with the ElGamal, and the Okamoto-Uchiyama [OU98] cryptosystems. In the following, we describe previous results and related work available in the literature.

2 Related Work

Much research has gone into the problem of managing personal data in accordance with a user-defined privacy policy. In [GMM06], for instance, Golle *et al.* propose a mechanism by which data collectors can be caught and penalized if they violate an agreed-upon policy, and disclose sensitive data about a data-subject. The main idea there is that a data-collector would place a *bounty*, which it must forfeit if a privacy violation is uncovered. The bounty could be explicit in the form of a bond, or implicit in the form of penalties imposed if privacy is violated. This technique however is geared towards violation detection after the fact, and assumes the existence of active *bounty hunters* who seek to induce dishonest data collectors into committing unlawful disclosures.

Another related approach is that of policy-based encryption by Bagga *et al.* [BM05, BM06]. Policy-based encryption allows a user to encrypt a message with respect to a credential-based policy, formalized as a monotone boolean expression. The encryption is such that only a user having access to a qualified set of credentials, complying with the policy, is able to successfully decrypt the message. The context in [BM05, BM06], however, is different from the one in this paper, since the goal there is to allow the user to send a secret message to a designated set of players defined by a policy. In our context, the user's data is already stored in a database, and the goal is to allow user-authorized parties to retrieve the user's data, without the database manager learning which data has been retrieved or the identity of the data subject. It is also not clear how revocability can be implemented in the context of [BM05, BM06].

In [SWP00], Song *et al.* present a scheme for the problem of searching keywords on encrypted data. The setting there consists of a user, and a server storing encrypted data owned by the user. The server can process search queries on the user's stored ciphertext, only if given proper authorization from the user. The proposed scheme also supports hidden user queries, where the server conducts the search without learning anything about the content of the query. Although somewhat related to our context, it is not clear how the work in [SWP00] can be applied to the problem we describe in this paper, since delegating querying capabilities to a third party, may require the user to reveal his encryption key, and thus share all of his past and future secrets. Besides, it is not clear how the identity of the data-owner can be hidden from the server, or how to impose (e.g., time or usage) restrictions on search capabilities delegated to a third party.

Finally, in [AIR01] Aiello *et al.* consider a scenario where a database contains a set of priced data items, and users privately retrieve data from it. The proposed protocol is called priced oblivious transfer, and allows a user U, who made an initial deposit, to buy different data items, without the database manager learning which items U is buying, as long as U's balance contains sufficient funds. We believe the construction in [AIR01] is the first to impose additional requirements on oblivious transfer protocols. While interesting in their own right, the added requirements do not address the identity of the data-subjects, and hence do not seem to help in solving the problem we consider in this paper.

3 Summary of Contribution and Paper Organization

We propose a solution to the accredited SPIR problem, and three constructions to implement it. The solution we present allows a user to issue authorizations to a Receiver to privately retrieve his records, without the database manager learning anything about the retrieved data, or the data subject's identity. The authorizations contain computationally non-modifiable, unforgeable, user-defined policies and limitations, governing their use. The authorizations can also be anonymously revoked by their issuer if needed.

To the best of our knowledge, this work is the first to give a solution to the accredited SPIR problem, and to address the more general issue of enforcing user-defined privacy policies, by combining SPIR protocols and privacy-preserving digital credentials.

The remainder of this paper is organized as follows. In Section 4, we describe the main building blocks used in the first construction and throughout the paper. In Section 5, we present a DL-based accredited SPIR construction. In Sections 6 and 7, we discuss the security, privacy features, and performance of the first construction. In Section 8.2, we present a second construction based on a RSA version of Brands credentials. In Section 8.4, we give a third variant based on the Okamoto-Uchiyama cryptosystem. We conclude in Section 9.

4 Building Blocks for the DL-Based Construction

4.1 Brands-CP Credentials

In [Bra00], Brands proposes various credential systems based on the hardness of the discrete logarithm problem in groups of prime order, and the RSA problem in groups of composite order (RSA groups). Brands has also proposed other variants of the above systems, based on DSA and the Chaum-Pedersen signatures [Bra00, Section 4.5.2]. For the purpose of our first construction, we will use the latter variant, and will refer to it as the Brands-CP system. The security of the Brands-CP system is based on the hardness of the discrete logarithm problem in groups of prime order. The Brands-CP system allows a certification authority CA to issue to a user U a set of credentials encoding attributes about U's identity. The credential itself consists of (1) a public key h embedding the user's attributes and (2) a special CA-supplied digital signature on it, denoted $\sigma_{\text{CA}}(h)$. At the end of the issuing protocol, the credential that user U has obtained is perfectly hidden from the CA, and perfectly indistinguishable from any other credential the CA has previously issued. Later, user U can show his credential to individuals and organizations in return for a service. Showing a credential does not necessarily require the revealing of the attributes encoded in it. A credential holder can selectively and verifiably disclose any information he wishes about his attributes, which may include revealing the actual values of the attributes, or just proving a predicate about them. In [Bra00, Section 3.6], Brands shows how to prove a class of linear predicates about the attributes. At a later stage, and depending on the application, the verifying individual or organization may want to deposit the credential showing transcript to the certification authority. This deposit can be thought of as a cheque deposit in the context of e-banking or as a ballot submission in the context of e-voting. The deposited transcript is unlinkable to the instance of the issuing protocol that generated the credential. For the sake of completeness, we give in the following a brief description of the issuing, and showing protocols of the Brands-CP system, as well as an overview of the parameters and setting.

System setting. On input the security parameter κ, the CA chooses κ-sized primes p and q such that $2q|p-1$. Let G_q be the unique subgroup of $\mathbb{Z}_p^*$ of order q, and let g_0 be one of its generators. The CA also chooses $H : \{0,1\}^* \to \mathbb{Z}_q^*$, a public collision-resistant hash function. In the setup phase, the certification authority randomly chooses $y_1, y_2, \cdots, y_\ell$ and $x_0 \in_R \mathbb{Z}_q^*$, and computes $(g_1, g_2, \cdots, g_\ell, h_0) := (g_0^{y_1}, g_0^{y_2}, \cdots, g_0^{y_\ell}, g_0^{x_0}) \mod p$. The parameters $(g_1, g_2, \cdots, g_\ell, h_0)$ are then made public along with g_0, q, and G_q.

Credential issuing. To obtain a credential, a user first convinces the certification authority that he fulfills a set of application-specific requirements necessary to receive that credential. The certification authority then encodes a set of ℓ of user attributes in the credential. Let $x_1, \cdots, x_\ell$ denote the attributes to be encoded. The credential's public key is then computed as $h := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^\alpha$, where α is a secret blinding factor randomly chosen in $\mathbb{Z}_q^*$ by the user. The certification authority's signature on the credential is a triplet $(c_0', r_0', z') \in \mathbb{Z}_q^2 \times G_q$, satisfying the relation $c_0' = H(h, z', g_0^{r_0'} h_0^{-c_0'}, h^{r_0'} z'^{-c_0'})$. At the end of the issuing protocol, the certification authority knows neither h nor the signature (c_0', r_0', z').

Credential showing. In order to have access to a service, user U can show his credential without the verifying party being able (1) to learn information about the encoded attributes beyond what U willingly discloses, or (2) to link the credential to the user's identity even if it colludes with the certification authority. In practice, to show his credential to a verifying party, user U reveals (1) the credential's public key h along with a signature $\sigma_{\mathrm{CA}(h)} := (z', c_0', r_0')$, and (2) a signed proof of knowledge of a representation of h, with respect to basis $(g_1, g_2, \cdots, g_\ell, h_0)$. This signature is performed on a verifier-chosen challenge m. The verifier checks the validity of the credential by verifying if the relation $c_0' \overset{?}{=} H(h, z', g_0^{r_0'} h_0^{-c_0'}, h^{r_0'} z'^{-c_0'})$ holds. If the credential is valid, the verifier moves on to check the validity of the signed proof of knowledge.

More details on the Brands-CP credential issuing and showing protocols, as well as their security and privacy properties, can be found in [Lay07].

4.2 ElGamal Homomorphic Encryption

Our DL-based accredited SPIR construction relies on the ElGamal encryption scheme because of its homomorphic properties and because it fits well the setting of the Brands-CP credentials. In the following we recall the settings of the ElGamal cryptosystem.

Settings. Let p, q, and G_q, be the public parameters chosen by the CA in the setup of the Brands-CP credential system. User U randomly chooses g_{ElG}, a generator of G_q, and $x_u \in_R \mathbb{Z}_q^*$, and computes $y_{\mathsf{ElG}} := g^{x_u} \mod p$. User U then publishes his ElGamal public key $(G_q, g_{\mathsf{ElG}}, y_{\mathsf{ElG}})$, and keeps his private key x_u secret. A message $m \in G_q$, can be encrypted by choosing a random $r \in_R \mathbb{Z}_q^*$, and computing $c = (g_{\mathsf{ElG}}^r, y_{\mathsf{ElG}}^r m) = (c_1, c_2)$. Using U's private key, the plaintext can be recovered as $m = c_2/c_1^{x_u}$. Given a constant α, and encryptions of m and m', it is easy to compute randomized encryptions of $m \times m'$ and m^α.

4.3 AIR-Based Lipmaa $\binom{1}{n}$-OT

In [Lip05], Lipmaa proposes a SPIR scheme based on ideas from a construction by Aiello et al. [AIR01]. Lipmaa's SPIR scheme is computationally private for the Receiver and perfectly private for the Sender. Its security relies on the hardness of the *decisional Diffie-Hellman* and the *decisional composite residuosity* problems [Lip05]. The SPIR scheme in [Lip05] has a log-squared communication complexity (in the size of the database).

Main idea. Let DB denote the Sender's private database, and let s be the index of the record the Receiver is interested in. The receiver computes $\mathsf{c} := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(s)$, a homomorphic encryption of s, and sends it to the Sender. Using the homomorphic properties of the encryption, the Sender computes for each record $\mathrm{DB}[j]$ in the database, $\mathrm{DB}'[j] := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_j(s-j) + \mathrm{DB}[j])$, where δ_j is a random blinding factor chosen by the Sender. The encrypted records $\mathrm{DB}'[j]$ are then sent to the Receiver, who will be able to retrieve something meaningful only from $\mathrm{DB}'[s] := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\mathrm{DB}[s])$; everything else will decrypt to randomness. The construction in [Lip05] follows a similar methodology to the above, except that the Sender uses an extra loop of superposed encryptions that leads to a randomized ciphertext of $\mathrm{DB}'[s]$. Only the latter is sent back to the Receiver. This is done as follows. The database $(\mathrm{DB}[1], \cdots, \mathrm{DB}[n])$ is arranged in an α-dimensional $\lambda_1 \times \cdots \times \lambda_\alpha$ hyper-rectangle for some pre-defined positive integers λ_j, such that $n = \prod_{j=1}^{\alpha} \lambda_j$. Each record $\mathrm{DB}[i]$ is indexed by a tuple $(i_1, \cdots, i_\alpha)$ on this hyper-rectangle, where $i_j \in \mathbb{Z}_{\lambda_j}$. To retrieve a particular record $(s_1, \cdots, s_\alpha)$, the Receiver submits to the Sender a homomorphic encryption $\beta_{jt} := HomEnc_{pk}(b_{jt})$, for $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$, where $b_{jt} = 1$ if $t = s_j$, and $b_{jt} = 0$ otherwise. The Sender exploits the homomorphic properties of the encryption scheme $HomEnc_{pk}(.)$ to create a new $(\alpha - 1)$-dimensional database DB_1, such that $\forall (i_2, \cdots, i_\alpha) \in \mathbb{Z}_{\lambda_2} \times \cdots \times \mathbb{Z}_{\lambda_\alpha}$, $\mathrm{DB}_1(i_2, \cdots, i_\alpha)$ is equal to an encryption of $\mathrm{DB}_0(s_1, i_2, \cdots, i_\alpha)$, where DB_0 is the the Sender's original database DB. The same procedure is repeated, and at the j^{th} iteration, an $(\alpha - j)$-dimensional database DB_j is obtained by the Sender, such that $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha)$ is equal to a j-times encryption of $\mathrm{DB}_0(s_1, \cdots, s_{j-1}, i_j, \cdots, i_\alpha)$. After α iterations, the Sender obtains DB_α, an α-times encryption of $\mathrm{DB}_0(s_1, \cdots, s_\alpha)$. The Sender returns DB_α to the Receiver, who needs to decrypt it α times to recover $\mathrm{DB}(s)$. In [Lip05], Lipmaa uses a special *Length Flexible Additively Homomorphic* encryption scheme to implement $HomEnc_{pk}(.)$. Notice that in the hyper-rectangle construction above, the Receiver can cheat by maliciously sending $\beta_{jt} := HomEnc_{pk}(1)$, for $t \neq s_j$. To stop such attacks, the Sender performs the repeated encryptions above, on $\mathrm{DB}_0 = \mathrm{DB}'$ rather than on the Sender's original DB. Let s' be the index corresponding to the β_{jt}'s. At the end of the protocol, the Receiver obtains DB_α, which he decrypts α times to recover $\mathrm{DB}'[s'] = \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_{s'}(s-s') + \mathrm{DB}[s'])$. Next, the Receiver decrypts $\mathrm{DB}'[s']$ once again, and recovers something meaningful ($\mathrm{DB}[s]$) only if $s' = s$. More details can be found in [Lay07].

Remark. Both [Lip05] and [AIR01] propose the ElGamal cryptosystem to implement $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}$. It is worth noting however, that using plain ElGamal, it is not possible to compute $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_j(s-j) + \mathrm{DB}[j])$ given $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(s)$, since ElGamal is *only multiplicatively* homomorphic. We fix this problem in the next section.

5 Accredited SPIR Based on the DL Problem

The DL-based accredited SPIR scheme we propose, is achieved by combining the three building blocks above, modulo few adaptations. We first give a high-level overview of the construction, before getting into the details. We assume the public parameters of the three building blocks are already known to all parties. Let $\mathrm{ID}_\mathcal{A}$ be an attribute, that uniquely identifies the Authorizer (e.g., an SSN). This $\mathrm{ID}_\mathcal{A}$ will determine the index by which the Receiver will query the Sender's database. Let us first assume that the Authorizer possesses a Brands-CP credential of the form $(h, \sigma_{\mathrm{CA}}(h))$, where $h = (g_1^{\mathrm{ID}_\mathcal{A}} g_2^{x_2} \cdots g_\ell^{x_\ell} h_0)^\alpha$. The Authorizer computes $\mathsf{c} := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}((g_{db})^{\mathrm{ID}_\mathcal{A}}) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\mathrm{ID}_\mathcal{A}}) := (\mathsf{c}_1, \mathsf{c}_2)$, where $\mathsf{pk}_{\mathsf{ElG}}$ is the Receiver's ElGamal public key, and g_{db} is a public generator of G_q chosen by the Sender. Next, the Authorizer produces a signed proof of knowledge asserting that the logarithm, to the base g_{db}, of the plaintext encoded in c, is the same as the first attribute embedded in credential h. We call this last assertion an ID-consistency proof.

Notice that this latter proof cannot be done in a straightforward way using the original Brands-CP credentials, because h has the form $h := g_1^{\beta_1} \cdots g_\ell^{\beta_\ell} h_0^\alpha$, where $\beta_i = \alpha x_i$ for some random blinding factor α. Establishing ID-consistency in this case requires proving a non-linear predicate on secret exponents $(\alpha, \beta_1, \mathrm{ID}_\mathcal{A})$, defined by $\mathcal{P} \equiv$ "$\beta_1 = \alpha \times \mathrm{ID}_\mathcal{A}$", which cannot be done efficiently. To fix this problem we propose a modified version of the Brands-CP credentials, with exactly the same security and privacy properties. In the modified version, the credential's public key h is computed as $h := (g_1^{x_1} \cdots g_{\ell-1}^{x_{\ell-1}} g_\ell^\alpha h_0)$, where $x_1, \cdots, x_{\ell-1}$ are identity attributes, and α is a secret random blinding factor chosen by the credential recipient. This modification is of general interest, and can be used in other contexts as well. For lack of space, a summary of the modified credential system is outlined in [Lay07].

In what follows, we assume the Authorizer possesses a credential of the new type. Let us denote the public key of the Authorizer's new credential by $h := (g_1^{\mathrm{ID}_\mathcal{A}} \cdots g_{\ell-1}^{x_{\ell-1}} g_\ell^{\alpha_1} h_0)$. To prove ID-consistency between h and the SPIR query

$(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$, it suffices for the Authorizer to produce a signed proof of knowledge of a DL-representation of $(h/\mathsf{c}_2 \mod p)$ with respect to basis $((g_1 g_{db}^{-1}), g_2, \cdots, g_\ell, h_0, g_{\mathsf{ElG}}, y_{\mathsf{ElG}})$. We denote the latter signed proof by $\mathrm{SPK}\{(\varepsilon_1, \cdots, \varepsilon_\ell, \mu, \nu) : h = g_1^{\varepsilon_1} \cdots g_\ell^{\varepsilon_\ell} h_0 \wedge \mathsf{c}_2 = y_{\mathsf{ElG}}^{\mu} g_{db}^{\nu} \wedge \varepsilon_1 = \nu\}(m)$. The message m to be signed, can be a concatenation of several fields, including a fresh nonce. In addition, it may contain the identity of the Receiver, which will allow the Authorizer to exclusively tie the authorization to the Receiver, and discourage him from sharing it with a third party. The Authorizer may also include an expiry date in m to make sure his authorization remains valid only for the appropriate amount of time. More generally, the Authorizer may encode in m any application-specific policy he wants the Receiver to follow.

Running the SPIR. Now let us assume the signed proof above was accepted. The next step, would be for the Receiver to compute the query messages as indicated in the OT scheme of Section 4.3. First let $(\mathrm{ID}_{\mathcal{A}}{}^{(1)}, \cdots, \mathrm{ID}_{\mathcal{A}}{}^{(\alpha)})$ be the representation of the Authorizer's $\mathrm{ID}_{\mathcal{A}}$ in the α-dimensional hyper-rectangle $\lambda_1 \times \cdots \times \lambda_\alpha$ used by the Sender's database. The Receiver then computes, for $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$, the homomorphic encryptions $\beta_{jt} := HomEnc_{pk}(b_{jt})$, where $b_{jt} = 1$ if $t = \mathrm{ID}_{\mathcal{A}}{}^{(j)}$, and $b_{jt} = 0$ otherwise. Next, the Receiver submits to Sender: (1) the credential $(h, \sigma_{\mathrm{CA}}(h))$, (2) the first part of the query $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$, (3) an ID-consistency proof $\mathrm{SPK}\{(\varepsilon_1, \cdots, \varepsilon_\ell, \mu, \nu) : h = g_1^{\varepsilon_1} \cdots g_\ell^{\varepsilon_\ell} h_0 \wedge \mathsf{c}_2 = y_{\mathsf{ElG}}^{\mu} g_{db}^{\nu} \wedge \varepsilon_1 = \nu\}(m)$, and (4) the second part of the query consisting of the β_{jt}'s for $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$.

Note that there is no need for the Receiver to prove consistency between the β_{jt}'s and $(\mathsf{c}_1, \mathsf{c}_2)$. As we will show later, any attempt by the Receiver to incorrectly compute the β_{jt}'s, will prevent him from learning anything meaningful at the end of the SPIR protocol.

Once the ID-consistency check succeeds, the Sender starts processing the Receiver's query as explained in the following. But first, we make few practical assumptions. We assume that n, the size of the Sender's database, is bounded above by q, the order of G_q. In practice, q is chosen to be at least 160-bit long, which means the Sender's database could have up to 2^{160} different records. Although we think this should be sufficient in practice, the size of q can always be increased if needed. Moreover, we assume that each record $\mathrm{DB}[i]$ contains a field for storing $(g_{db}^{-i} \mod p)$, in addition to a large field containing application-specific data (e.g., health, financial data).

Now the query is processed as follows. Using the first part of the query, and the *multiplicatively* homomorphic properties of ElGamal, the Sender computes for $j \in [1, n]$, $\mathrm{DB}_0[j] = \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\delta_j(\mathrm{ID}_{\mathcal{A}} - j)} \times \mathrm{DB}[j]) = ((\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(g_{db}^{\mathrm{ID}_{\mathcal{A}}}) \times g_{db}^{-j})^{\delta_j} \times \mathrm{DB}[j])$, where δ_j is a random blinding factor chosen by the Sender. Note that g_{db}^{-j} has already been precomputed and stored in with DB's j^{th} record. The Sender then proceeds with computing DB_α by repeated encryptions of the records of DB_0 as indicated in the OT scheme of Section 4.3. Upon receiving DB_α, the Receiver recovers $\mathrm{DB}_0[\mathrm{ID}_{\mathcal{A}}]$ by repeated decryption using his secret homomorphic key sk. Next the sender obtains the desired record $\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}]$ by decrypting

$\mathrm{DB}_0[\mathrm{ID}_{\mathcal{A}}]$ using his ElGamal private key. A figure summarizing the protocol can be found in [Lay07].

6 Security and Privacy Properties

Theorem 1. *Assuming the DL problem is hard, the ElGamal and $HomEnc(.)$ cryptosystems IND-CPA secure, and that the none of the three parties colludes with the other, the protocol of Section 5 solves the Accredited SPIR problem in the random oracle model, and provides computational privacy for both the Authorizer and Receiver, and perfect privacy for the Sender.*

Proof Sketch.[1] *Correctness.* easy to check and relies on the homomorphic properties of the ElGamal and the $HomEnc(.)$ cryptosystems.

Soundness. Assume the Receiver does not follow the protocol, and maliciously uses an index $s \neq \mathrm{ID}_{\mathcal{A}}$ in the second part of the query. This will lead DB_α computed by the Sender to be an α-time encryption under $HomEnc_{pk}(.)$ of $DB_0[s] = \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\delta_s(\mathrm{ID}_{\mathcal{A}}-s)} \times \mathrm{DB}[s])$, where δ_s is a secret blinding factor. $DB_0[s]$ decrypts to random, which results in perfect privacy for the Sender.

Privacy for the Receiver. The Sender's view consists of a credential, an ElGamal encryption, a signed proof of knowledge, and a $HomEnc(.)$ encryption of an α-dimensional coordinate. Because the proof of knowledge is honest-verifier zero-knowledge for any distribution of the attributes (cf. [Bra00, Prop. 3.3.4]), seeing the signed proof of knowledge, together with the credential, does not leak any information to the Sender about the content of the Receiver's query. The Sender could only hope to extract information from the ElGamal ciphertext c and the homomorphic encryptions β_{jt}'s. But this should not be possible, because it implies breaking the security of ElGamal, or the $HomEnc(.)$ cryptosystems, which contradicts our assumption.

Privacy for the Authorizer. Assuming the Receiver and the Sender do not collude, there are two ways to violate the Authorizer's privacy. Either by learning information from the Receiver's query, or by forging a signed proof of knowledge on behalf of the Authorizer. The former attack is computationally impossible and follows from the Receiver's privacy. To achieve the second however, one could either (1) forge a new credential from scratch with the Authorizer's identity embedded in it, (2) forge a signed proof on a legitimately-obtained credential that was not issued to the Authorizer, and thus does not initially contain his identity, or (3) forge a signed proof on behalf of the Authorizer on a credential owned by the Authorizer. Attack (1) can be ruled out based on the computational unforgeability of the Brands-CP credentials in the random oracle model (cf. [Bra00, Prop. 4.3.7]). Similarly, attack (2) is impossible because of the *non-modifiability* property of Brands-CP credentials [Bra00, Prop. 3.3.8].

[1] Complete details will be given in the full version of the paper.

The non-modifiability feature states that, assuming the DL problem is hard, it is infeasible, in the random oracle model, to construct a signed proof for a formula that does not in fact apply to the prover's representation. Finally, attack (3) can be ruled out as well, owing to the unforgeability of signed proofs on Brands-CP credentials [Bra00, Prop. 3.3.6].

6.1 Additional Privacy for the Authorizer

User-centricity and policy enforcement. When issuing an authorization, the credential holder could specify in the message of the signed proof of knowledge, a set of rules that he wishes the authorization recipient to comply with. For instance, he could specify an expiry date, an upper bound on the number of times the authorization is used, or any other usage policy. The Sender is supposed to refuse processing queries from a Receiver who does not satisfy the usage policy specified in the signature.

Revocability. The Authorizer may decide to revoke a previously issued authorization. This can be done anonymously as follows. The Authorizer first needs to prove knowledge, over an physically anonymized channel (e.g., a MixNet [Cha81]), of a representation of the credential used in the authorization to be revoked. Once the proof of knowledge is accepted, the Authorizer requires the Sender (DB manager) to add the credential in question to a black list of revoked authorizations. Later, it is easy for the Sender to check whether the credential contained in a submitted query is on the black list or not. This can be done efficiently using hash tables for instance. Note that an authorization can be revoked only by its issuer, since an polytime adversary cannot find a discrete-log representation with non-negligeable probability.

Authenticated personal information retrieval. In the special case, where the Authorizer and Receiver are the same entity, the construction we propose provides the data-subject with a mechanism to retrieve *his own* personal data anonymously. Our construction also ensures that the stored data can be retrieved only by its owner. The channel between the Receiver and Sender in this case has to be physically anonymized.

7 Performance Analysis

The accredited SPIR construction of Section 5 does not lead to a significant increase in computation and communication complexity, compared to the underlying SPIR scheme [Lip05]. If we assume the Authorizer has a credential with $(\ell - 1)$ attributes, then the added computation complexity is as follows. The Authorizer needs to make $(\ell + 6)$ offline exponentiations (all precomputable), while the Receiver and Sender, both need to make $(\ell + 8)$ online exponentiations. This is negligeable compared to the complexity of the underlying SPIR scheme which is linear in n, the size of the database. In practice, ℓ is in the range of 20, while $n \approx 2^{160}$. In terms of communication complexity, both the

Authorizer and Receiver need to send $(\ell + 8)\log(n) + 5$ extra bits to the Receiver and Sender respectively. Again this does not change the overall $\mathcal{O}\big(\log^2(n)\big)$ asymptotic communication complexity of the underlying SPIR scheme [Lip05].

8 Accredited SPIR Based on RSA

The constructions we present in this section are based on a RSA-version of Brands' credentials [Bra00, Section 4.2.2]. For the sake of completeness, we briefly introduce them in the following.

8.1 Brands-RSA Credentials

Settings. On input the security parameter κ, the credential issuer chooses: (1) κ-sized primes P and Q, and computes $N := PQ$, (2) a prime v smaller than N, and co-prime to $\phi(N)$, (3) random elements $(g_1, \cdots, g_\ell) \in_R (\mathbb{Z}_N^*)^\ell$, and (4) a one-way hash function $\mathcal{H}(.) = \mathcal{H}_{N,v}(.) : \{0,1\}^* \to \mathbb{Z}_s$, for some s superpolynomial in κ.

The credential issuer makes the parameters $N, v, (g_1, \cdots, g_\ell), \mathcal{H}$ public, and keeps P and Q secret. In addition, the issuer chooses $x_0 \in_R \mathbb{Z}_v^*$, such that given $h_0 := x_0^v \mod N$, computing the v^{th} root of h_0 is hard. The issuer then publishes h_0 and keeps x_0 secret.

Credential issuing. Assume after making the necessary identity checks, the certification authority accepts to issue a credential to the user. Let $(x_1, \cdots, x_\ell) \in (\mathbb{Z}_v^*)^\ell$ be the attributes the CA wants to encode in the credential, and let $h := g_1^{x_1} \cdots g_\ell^{x_\ell} \mod N$. The x_i's are known to both the user and the CA. The user then chooses a random blinding factor $\alpha_1 \in \mathbb{Z}_N^*$ and computes the credential's public key $h' := h\alpha_1^v$. The certification authority's digital signature on the credential is a pair $(c_0', r_0') \in \mathbb{Z}_s \times \mathbb{Z}_N^*$, satisfying the relation $c_0' = \mathcal{H}(h', {r_0'}^v (h_0 h')^{-c_0'})$. At the end of the issuing protocol, the certification authority knows neither h' nor the signature (c_0', r_0').

Credential showing. Similar to the Brands-CP system, a user can show his credential to a verifying party, by first revealing the credential's public key h' and CA-signature (c_0', r_0'). The verifier checks if the validity relation $c_0' \overset{?}{=} \mathcal{H}(h', {r_0'}^v (h_0 h')^{-c_0'})$ holds. Once the validity check succeeds, the user produces a signed proof of knowledge of a RSA representation $(x_1, \cdots, x_\ell, \alpha_1)$ of h' with respect to basis the $(g_1, \cdots, g_\ell, v)$. The signed proof can also be computed with respect to a predicate $\mathcal{P}$ on exponents $(x_1, \cdots, x_\ell)$, agreed-upon by the user and the verifier at the time of the showing.

More details on the Brands-RSA credential issuing and showing protocols can be found in [Lay07].

8.2 Combining ElGamal Encryption with Brands RSA-Based Credentials

We assume the Authorizer possesses a Brands-RSA credential h', and certificate $\sigma_{\text{CA}}(h') := (c_0', r_0')$, with h' of the form $h' = (g_1^{\text{ID}_\mathcal{A}} g_2^{x_2} \cdots g_\ell^{x_\ell} \alpha^v) \mod N$. Recall that $\mathrm{ID}_{\mathcal{A}}, x_2 \cdots x_\ell$ are elements of $\mathbb{Z}_v$, and that $\mathrm{ID}_{\mathcal{A}}$ represents a record index in the Sender's DB. To accommodate all possible DB indexes $\mathrm{ID}_{\mathcal{J}}$ (ranging over $[1, n] = [1, q]$), the prime v is chosen to be greater than q, the order of G_q in the ElGamal setting. Also, for reasons that will become clear shortly, the prime v is chosen to be coprime to $p - 1$, where p denotes the public prime parameter in the ElGamal setting. Finally, the RSA factors P and Q are chosen to be greater than p.

As in the first construction based on Brands-CP credentials, the Authorizer (data subject) and Receiver use the ElGamal cryptosystem to compute the SPIR query. Assuming the same setting for the Sender and Receiver as in Section 5, the Authorizer computes $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$. To prove ID-consistency between h' and the SPIR query, it suffices for the Authorizer to produce a signed proof of knowledge of a RSA-representation of $(h'/\mathsf{c}_2 \mod Np)$ with respect to basis $((g_1 g_{db}^{-1}), g_2, \cdots, g_\ell, \mathsf{y}_{\mathsf{ElG}}^{-1}, v)$.

Putting the pieces together. As in the Brands-CP case, the Authorizer proves ID-consistency between his credential and the query by sending to the Receiver a signed proof of knowledge of the form $\mathrm{SPK}\{(\varepsilon_1, \cdots, \varepsilon_{\ell+1}, \mu) : h'/\mathsf{c}_2 = (g_1 g_{db}^{-1})^{\varepsilon_1} g_2^{\varepsilon_2} \cdots g_\ell^{\varepsilon_\ell} (\mathsf{y}_{\mathsf{ElG}}^{-1})^{\varepsilon_{\ell+1}} \mu^v \mod Np\}(m)$. The Authorizer can use the message m to encode any usage policy he wants the Receiver to follow.

After accepting the signed proof, the Receiver proceeds with the SPIR protocol of Section 5 without any further changes.

8.3 Security and Privacy Properties

Theorem 2. *Assuming the (multi-prime) RSA problem is hard, the ElGamal and HomEnc(.) cryptosystems IND-CPA secure, and that none of the three parties colludes with the other, the protocol of Section 8.2 solves the Accredited SPIR problem in the random oracle model, and provides computational privacy for both the Authorizer and Receiver, and perfect privacy for the Sender.*

Complete details of the proof will be given in the full version of the paper. The properties of *user-centricity*, *revocability*, and *authenticated PIR* described in Section 6.1, do apply for the new scheme as well.

8.4 Variant Based on the Okamoto-Uchiyama Cryptosystem

The construction of Section 8.2, can be modified by using the Okamoto-Uchiyama cryptosystem [OU98] instead of ElGamal. The Okamoto-Uchiyama cryptosystem is a probabilistic public key cryptosystem whose security is equivalent to the problem of factoring moduli of the form $n = p^2 q$, for p and q prime. The Okamoto-Uchiyama cryptosystem is additively homomorphic.

Setting of the Okamoto-Uchiyama cryptosystem. Given security parameter κ, choose κ-sized primes p and q, and let $n = p^2 q$. The choice of p and q should be such that $gcd(p, q - 1) = gcd(q, p - 1) = 1$. Choose random $g \in \mathbb{Z}_n^*$, such that $g_p = g^{p-1} \mod p^2$ has order p. Let $h = g^n \mod n$. The tuple (n, g, h, κ)

is published as the public key, while (p, q) are kept secret.To encrypt a message $0 < m < 2^{\kappa-1}$, select random $r \in \mathbb{Z}_n$, and compute $\mathsf{E}_{\mathsf{pk}_{\mathsf{OU}}}(m, r) := g^m h^r \mod n$. The decryption function uses a special "logarithmic function" [OU98].

Putting the pieces together. The Authorizer uses the Receiver's public key (in this case the Okamoto-Uchiyama public key) to produce the SPIR query and prove ID-consistency between the latter and an Authorizer's credential. Let $\mathsf{c} := \mathsf{E}_{\mathsf{pk}_{\mathsf{OU}}}(\mathrm{ID}_{\mathcal{A}}, r) = g^{\mathrm{ID}_{\mathcal{A}}} h^r \mod n$ be a randomized encryption of the Authorizer's ID. Moreover, let $(h', \sigma_{\mathrm{CA}}(h'))$ be the Authorizer's Brands-RSA credential, with $h' = (g_1^{\mathrm{ID}_{\mathcal{A}}} g_2^{x_2} \cdots g_\ell^{x_\ell} \alpha^v) \mod N$. The Authorizer computes $h'/\mathsf{c} = ((g_1 g^{-1})^{\mathrm{ID}_{\mathcal{A}}} g_2^{x_2} \cdots g_\ell^{x_\ell} h^{-r} \alpha^v) \mod Nn$, and produces a signed proof of knowledge of a RSA representation of h'/c with respect to basis $((g_1 g^{-1}), g_2, \cdots, g_\ell, h^{-1}, v)$. With very high probability, parameters g and h are coprime with N, otherwise they can be used to factor N and break the security of the Brands-RSA credential system. Therefore, with very high probability, g^{-1} and h^{-1} exist modulo Nn. Similarly, with very high probability $gcd(v, \phi(Nn)) = 1$, otherwise v can be used to factor n and break the Okamoto-Uchiyama cryptosystem. Therefore, the RSA representation above is well defined. Once the signed proof is accepted, the Receiver deposits the SPIR query together with the signed proof to the Sender. The Sender in turn checks the validity of the credential, and signed proof, and proceeds with the remaining steps of the original SPIR scheme of Section 4.3.

9 Conclusion

The paper describes a new access control scheme, where access policies are defined by the data subjects. More specifically, the proposed scheme allows database managers to be convinced that each of their stored data is being retrieved according to the policies of the data subjects, without learning which data has been retrieved or the identity of its owner. We present three constructions based on the discrete logarithm and RSA problems. The constructions we propose rely on anonymous authorizations, and combine SPIR systems and privacy-preserving digital credentials. The authorizations contain non-modifiable, unforgeable, user-defined policies governing their use. Moreover, authorizations can be anonymously revoked by their issuers. The proposed solutions yield only a negligeable increase in complexity compared to that of the underlying SPIR scheme. This work can be extended in a number of ways. For example it would be interesting to add a mechanism to support the "authorized and anonymous editing of records". One could also try to improve efficiency, and propose additional constructions based on other building blocks and assumptions.

Acknowledgement. The author is grateful to Stefan Brands for interesting discussions and for pointing out related literature. This work is supported by the IWT SBO ADAPID project (Advanced Applications for e-ID cards in Flanders). The author was previously funded in part by the Universitary Mission of Tunisia in North America.

References

[AIR01] Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)

[BM05] Bagga, W., Molva, R.: Policy-based cryptography and applications. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 72–87. Springer, Heidelberg (2005)

[BM06] Bagga, W., Molva, R.: Collusion-free policy-based encryption. In: Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 233–245. Springer, Heidelberg (2006)

[Bra00] Brands, S.: Rethinking public key infrastructures and digital certificates: Building in privacy, The MIT Press,(2000) Available for download at http://www.credentica.com/the_mit_pressbook.html

[Cha81] Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–90 (1981)

[Cha85] Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)

[CL02] Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

[CL04] Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)

[CMO00] Di Crescenzo, G., Malkin, T., Ostrovsky, R.: Single database private information retrieval implies oblivious transfer. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 122–138. Springer, Heidelberg (2000)

[CP92] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)

[GIKM98] Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 151–160. ACM Press, New York (1998)

[GMM06] Golle, P., McSherry, F., Mironov, I.: Data collection with self-enforcing privacy. In: ACM Conference on Computer and Communications Security—ACM CCS 2006, pp. 69–78. ACM, New York (2006)

[KO97] Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: FOCS '97. Proceedings of the 38th Annual Symposium on Foundations of Computer Science, pp. 364–373. IEEE Computer Society, Los Alamitos (1997)

[Lay07] Mohamed Layouni, Controlling access to personal data through Accredited Symmetrically Private Information Retrieval, Tech. Report ??, McGill University, School of Computer Science,(July 2007)

[Lip05] Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)

[OU98] Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)

[SWP00] Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society, Los Alamitos (2000)

Generic Certificateless Encryption in the Standard Model*

Qiong Huang and Duncan S. Wong

Department of Computer Science,
City University of Hong Kong,
Hong Kong, China
{csqhuang,duncan}@cityu.edu.hk

Abstract. Despite the large number of certificateless encryption schemes recently proposed, many of them have been found to be insecure under a practical attack called *malicious-but-passive* KGC attack, since they all follow the same key generation procedure as that of the one proposed by Al-Riyami and Paterson in ASIACRYPT 2003. The only scheme that remains secure against this attack is due to Libert and Quisquater (PKC 2006). However, the security can only be shown in the random oracle model. In this paper, we first show that a scheme which has a different key generation procedure from that of Al-Riyami and Paterson also suffers from the malicious-but-passive KGC attack. Our attacking techniques are different and may cause greater extent of damage than the previous ones. We also propose a generic construction of certificateless encryption which can be proven secure against this attack *in the standard model*. This generic scheme not only is the first one proven secure in the standard model, but is also very efficient to instantiate. We also describe how to use short signature and hybrid encryption to construct highly efficient instantiations of this generic scheme.

1 Introduction

In traditional public key cryptography, a user selects a public/private key pair (pk, sk) and publishes pk. A *certificate*, which essentially is a signature on the user's identity and pk issued by a *certification authority* (CA), will then be employed for indicating the relationship between the user and pk. This method works under the public key infrastructure (PKI) involves a lot of additional work for managing the certificates that include revocation, storage and distribution.

In 1984, Shamir [18], introduced the notion of identity-based cryptography, aiming to alleviate the existing problems in PKI by getting rid of certificates. A user can use an email address, an IP address or any other information related to his identity, that is publicly known and unique in the whole system, as his public key. There is a trusted party, called Key Generation Center (KGC), which is in charge of the user private key generation. The advantage of an identity-based

* The work was supported by grants from CityU (Project Nos. 7001844, 7001959 and 7002001).

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 278–291, 2007.
© Springer-Verlag Berlin Heidelberg 2007

cryptosystem is that anyone can simply use the user's identity to encrypt messages. This can be done even before the user gets its private key from the KGC. However, the user must also completely trust the KGC, which can impersonate the user and decrypt any of the user's ciphertexts. This issue is generally referred to as key escrow problem in identity-based cryptography.

In 2003, Al-Riyami and Paterson [1] introduced *certificateless cryptography*, which is intended to solve the key escrow problem that is inherent in identity-based cryptography, while at the same time, eliminate the use of certificates as in the traditional PKI. In a certificateless cryptosystem, the KGC is involved to issue a user *partial* key psk_{ID} for a user with identity ID. The user independently generates a user public/private key pair (upk_{ID}, usk_{ID}), and publishes upk_{ID}. A message will then encrypted under *both* upk_{ID} and the user's identity ID. To decrypt a ciphertext, the user must have the knowledge of both the user partial key psk_{ID} and secret key usk_{ID}. Knowing only one of them does not allow the recovery of the plaintext.

Related Work. Since the introduction of certificateless cryptography [1], there have been many schemes proposed [21,14,16,12,13,2,17]. The original definition of certificateless encryption [1] consists of seven algorithms. It has recently been simplified to five [12,13] and has shown to be more versatile than the original one. In this paper, we also adopt the five-algorithm simplified version for defining a certificateless encryption scheme.

Yum and Lee proposed a generic certificateless encryption scheme in [20] which has later been shown to be insecure under the model of [1] by Libert and Quisquater [16] and Galindo *et al.* [10] independently. In [16], the authors also proposed a generic certificateless encryption scheme. However, their scheme is only proven secure in the random oracle model, which is a heuristic method for showing the security of cryptographic schemes. The security may not preserve when the random oracle is replaced by a hash function, even if the scheme is reduced to some complexity (or number-theoretic) assumption. Recently, Liu *et al.* [17] proposed another certificateless encryption scheme which, to the best of our knowledge, is the first one in the standard model. However, as we will show in this paper, their scheme is insecure against *malicious-but-passive* KGC attack which is introduced as follows.

In [2], Au *et al.* considered another strong security model for certificateless cryptography, in which the user's trust on the KGC is further relaxed. By using the term introduced in [2], the KGC of a certificateless cryptosystem can be *malicious-but-passive*. This means the KGC can be malicious so that it may not follow the scheme specification for generating system parameters and master key, while it does not actively replace a user's public key or corrupt the user's secret key. The purpose of such a malicious-but-passive KGC is to compromise a user's secret key without being detected. Since the KGC does not need to replace the user's public key or compromise the user's machine for corrupting the user's secret key, in practice, it is very difficult to find out the occurrence of this attack. Consider a user who is the president of a country or an organization. It may have a great incentive for the KGC to launch the malicious-but-passive attack as ciphertexts for the president may contain some valuable and profitable information. The attack is practical, and the new security model would become much stronger and realistic than the old ones in this aspect if the attack is properly formalized.

Under the *malicious-but-passive* KGC attacking model, certificateless cryptosystems proposed in [1,14,15] have been shown to be insecure, since they all follow the same key generation procedure as that of [1]. The only provably secure certificateless encryption scheme against malicious-but-passive KGC attack currently available is due to Libert and Quisquater [16]. The proof is given in [2]. They showed that the IND-CPA-secure certificateless encryption scheme in [16] is also IND-ID-CPA secure under the malicious-but-passive KGC adversarial model. Then, by applying the transformation technique of [16], any IND-ID-CPA-secure scheme can be converted to an IND-ID-CCA2-secure one. However, the final scheme is only secure in the random oracle model. There is no concrete or generic construction of certificateless encryption which is proven secure against malicious-but-passive KGC attacks in the standard model.

Our Results. As a motivation, we first show that a recently proposed certificateless encryption scheme [17] also suffers from the malicious-but-passive KGC attack. As mentioned above, schemes in [1,14,15] are vulnerable to the malicious-but-passive KGC attack described in [2] as they all follow the same key generation procedure as that of [1]. However, the attacking technique of [2] does not apply to the scheme in [17] as the key generation procedure is different. We propose two malicious-but-passive KGC attacks against the scheme in [17]. The first attack causes the same extent of damage as the attack described in [2] against [1,14,15]. The second attack may cause greater impact to the system since in this attack, the malicious-but-passive KGC is able to decrypt ciphertexts which are for any user in the system without pre-selecting a target user to attack. Note that our attacks do not refute the security claims made in [17], since in their security model, it is assumed that the KGC can launch attacks only after honestly generating the master public/private key pair (and system parameters).

Then we propose the *first generic* certificateless encryption scheme which is proven secure against malicious-but-passive KGC attacks *in the standard model*. The idea of our construction is simple. It is constructed based on three well-formalized primitives and can be considered as a sequential encryption as in [21,2], with an additional signature-based mechanism to defend against attacks discussed in [16], but without relying on the assumption of random oracles. The construction is also efficient. We will describe how to use short signature [3] and hybrid encryption to implement highly efficient instantiations of this generic scheme.

It's worthy to note that, similar ideas were used by Dolev *et al.* [8] to construct non-malleable encryption schemes, and by Canetti *et al.* [6,4] to build public key encryption schemes from a weak variant identity-based encryption schemes.

Paper Organization. In Sec. 2, we define the certificateless encryption scheme and its security. In Sec. 3, we show that the certificateless encryption scheme in

[17] is vulnerable to some new *malicious-but-passive* KGC attacks. Our generic construction of secure encryption schemes and its security analysis are provided in Sec. 4. We also describe some instantiations of the generic scheme based on short signature and hybrid encryption. The paper is concluded in Sec. 5.

2 Definition and Adversarial Model

A certificateless encryption scheme [1,2] consists of five (probabilistic) polynomial-time (PPT) algorithms:

- $\mathsf{MasterKeyGen}$: On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a master public/private key pair (mpk, msk).
- $\mathsf{PartialKeyGen}$: On input msk and an identity $\mathsf{ID} \in \{0,1\}^*$, it generates a user partial key psk_{ID}.
- $\mathsf{UserKeyGen}$: On input mpk and a user identity ID, it generates a user public/private key pair $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$.
- Enc: On input mpk, a user identity ID, a user public key upk_{ID} and a message m, it returns a ciphertext c.
- Dec: On input a user private key usk_{ID}, a user partial key psk_{ID}, and a ciphertext c, it returns the plaintext m.

In practice, the KGC (Key Generation Center) performs the first two algorithms: $\mathsf{MasterKeyGen}$ and $\mathsf{PartialKeyGen}$. The master public key mpk is then published and it is assumed that everyone in the system can get a legitimate copy of mpk. It is also assumed that the partial key is issued to the corresponding user via a secure channel so that no one except the intended user can get it. Every user in the system also performs $\mathsf{UserKeyGen}$ for generating its own public/private key pair and publishes the public key. The correctness requirement is defined in the conventional way. We skip the details and refer readers to [1,2] for details.

Adversarial Model. There are two types of security for a certificateless encryption scheme, Type-I security and Type-II security, along with two types of adversaries, $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Adversary $\mathcal{A}_1$ models a malicious adversary which compromises the user private key usk_{ID} or replaces the user public key upk_{ID}, however, cannot compromise the master private key msk nor get access to the user partial key psk_{ID}. Adversary $\mathcal{A}_2$ models the malicious-but-passive KGC which controls the generation of the master public/private key pair, and that of any user partial key psk_{ID}. The following are five oracles which can be accessed by the adversaries.

- $\mathsf{CreateUser}$: On input an identity $\mathsf{ID} \in \{0,1\}^*$, if ID has not been created, the oracle runs $psk_{\mathsf{ID}} \leftarrow \mathsf{PartialKeyGen}(msk, \mathsf{ID})$ and $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{UserKeyGen}(mpk, \mathsf{ID})$. It then stores $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ into a list List[1]. In both cases, upk_{ID} is returned.

[1] Note that the list List is shared among all these five oralces.

- $\mathsf{RevealPartialKey}$: On input an identity ID, the oracle searches List for an entry corresponding to ID. If it is not found, $\perp$ is returned; otherwise, the corresponding psk_{ID} is returned.
- $\mathsf{RevealSecretKey}$: On input an identity ID, the oracle searches List for the entry of ID. If it is not found, $\perp$ is returned; otherwise, the corresponding usk_{ID} is returned.
- $\mathsf{ReplaceKey}$: On input an identity ID along with a user public/private key pair (upk', usk'), the oracle searches List for the entry of ID. If it is not found, nothing will be carried out. If $usk' = \perp$, the oracle sets $usk' = usk_{\mathsf{ID}}$. Then, it updates $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ to $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$.
- $\mathsf{Decryption}$: On input an identity ID and a ciphertext c, the oracle searches List for the entry of ID. If it is not found, $\perp$ is returned. Otherwise, it runs $m \leftarrow \mathsf{Dec}(psk_{\mathsf{ID}}, usk_{\mathsf{ID}}, c)$ and returns m. Note that the original upk_{ID} (which is returned by $\mathsf{CreateUser}$ oracle) may have been replaced by the adversary.

***Remark 1* (On the Decryption Oracle):** In the original adversarial model of certificateless encryption [1,16], it is required that the Decryption oracle should provide correct decryptions even after the user public key has been replaced by the adversary while the corresponding user secret key is *not known to the oracle*. We believe that the model is hardly realistic. Hence in this paper, we only require the Decryption oracle to perform the decryption task by using the current user keys. This also captures the case that the user public key is replaced by the adversary, but the user secret key remains the same. It is possible that the message m recovered from the ciphertext by using the current usk_{ID} is $\perp$.

***Remark 2* (On the Stronger Decryption Oracle Again):** It seems to be somewhat *self-contradictory* if we require a certificateless encryption scheme to be secure against both the adversaries with access to the stronger Decryption oracle and the *malicious-but-passive* KGC. Suppose that it is secure against Type-I adversaries with access to the stronger Decryption oracle. The game simulator, which also plays the role of the KGC, has to simulate the Decryption oracle for the adversary. In order to provide correct decryption without the user private key, the simulator usually needs to play tricks in setting the system parameters. However, in the same way, the *malicious-but-passive* KGC should also be able to compute the correct plaintext with a high probability, without knowing user private keys.

Let $\mathcal{C}$ be the game challenger/simulator, and $k \in \mathbb{N}$ be the security parameter. We denote by $\mathcal{A}$ the Type-I adversary (or the Type-II adversary) in the following game:

Game-I (or, Game-II) :

1. If this is **Game-I**, $\mathcal{C}$ runs $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(1^k)$, and then invokes $\mathcal{A}_1$ on input 1^k and mpk. If this is **Game-II**, $\mathcal{C}$ runs $\mathcal{A}$ on input

1^k, which returns a master public key mpk to $\mathcal{C}$. Note that $\mathcal{A}$ cannot make any oracle query at this stage[2].

2. In the game, $\mathcal{A}$ can issue $\mathsf{CreateUser}$, $\mathsf{RevealPartialKey}$, $\mathsf{RevealSecretKey}$, $\mathsf{ReplaceKey}$ and $\mathsf{Decryption}$ queries adaptively. If this is **Game-II**, the oracle $\mathsf{RevealPartialKey}$ is no more needed by $\mathcal{A}$ since it has the master secret key, and when $\mathcal{A}$ issues a query to $\mathsf{CreateUser}$ oracle, it has to additionally provide the user partial key psk_{ID}.
3. $\mathcal{A}$ submits two equal-length messages (m_0, m_1) along with a target identity ID^*. $\mathcal{C}$ then selects a random bit $b \in \{0,1\}$, computes a challenge ciphertext c^* by running $c^* \leftarrow \mathsf{Enc}(mpk, \mathsf{ID}^*, upk_{\mathsf{ID}^*}, m_b)$, and returns c^* to $\mathcal{A}$, where upk_{ID^*} is the user public key currently in List for ID^*.
4. $\mathcal{A}$ continues to issue queries as in step 2. Finally it outputs a bit b' as its guess for bit b.

$\mathcal{A}$ is said to win the game if $b' = b$, and (1) $\mathcal{A}$ did not query $\mathsf{RevealPartialKey}$ on ID^*, (2) $\mathcal{A}$ did not query $\mathsf{Decryption}$ on (ID^*, c^*), (3) if this is **Game-II**, $\mathcal{A}$ did not query $\mathsf{ReplaceKey}$ on ID^*. We denote by $\Pr[\mathcal{A}\ \mathsf{Succ}]$ the probability that $\mathcal{A}$ wins the game, and define the *advantage* of $\mathcal{A}$ in **Game-I** (or, **Game-II**) to be $\mathsf{Adv}_{\mathcal{A}} = \left|\Pr[\mathcal{A}\ \mathsf{Succ}] - \frac{1}{2}\right|$.

Definition 1. *A certificateless encryption scheme* CLE *is said to be* Type-I secure *(resp.* Type-II secure*) if there is no probabilistic polynomial-time adversary* $\mathcal{A}$ *which wins* **Game-I** *(resp.* **Game-II***) with non-negligible advantage.* CLE *is said to be* $\mathsf{IND\text{-}ID\text{-}CCA2}$ secure *if it is both* Type-I secure *and* Type-II secure*.*

3 Malicious-But-Passive KGC Attack

We describe two new malicious-but-passive KGC attacking techniques (under **Game-II**) to compromise schemes that follow the key generation procedure described in [17]. The techniques are different from that in [2], which is used to compromise schemes based on another type of key generation procedures [1,14,15].

We briefly describe the certificateless encryption scheme proposed in [17] to a certain extent that our attacking technique can be understood without referring to the complete description of the original scheme. In the $\mathsf{MasterKeyGen}$ of [17], the KGC first generates a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ such that each group has order p. Then, a generator g of $\mathbb{G}_1$ is selected. This is followed by the selection of a set of random elements in $\mathbb{G}_1$. The parameters we are going to use in the attack below are $g_2, g_1', h_1, u' \in_R \mathbb{G}_1$ and $\hat{U} = \{\hat{u}_i\}$ where $\hat{u}_i \in_R \mathbb{G}_1$, for $i = 1, \cdots, n$, and some $n \in \mathbb{Z}$. We skip the description of the remaining steps of $\mathsf{MasterKeyGen}$ and also the entire $\mathsf{PartialKeyGen}$. In $\mathsf{UserKeyGen}$, the user public key upk_{ID^*} for

[2] One exception is that if a scheme is analyzed under the random oracle model, $\mathcal{A}$ can query the random oracle. In this paper, we do not consider this exception as the security of our scheme proposed in the subsequent section (Sec. 4) will be shown in the standard model.

a user with identity ID^* is denoted by $(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}) \in \mathbb{G}_1^2$. We do not need to look into how these two elements are generated. Our first attack is described as follows.

Attack 1. The malicious-but-passive KGC (that is $\mathcal{A}_2$ in **Game-II**) arbitrarily selects a target identity ID^*. It computes $\mathfrak{u} = H_u(\mathsf{ID}^*)$, where $H_u : \{0,1\}^* \to \{0,1\}^n$ is a collision-resistant hash function pre-defined for this scheme. Let $\mathfrak{u}[i]$ be the i-th bit of $\mathfrak{u}$. Define $\mathcal{U} \subseteq \{1, \cdots, n\}$ to be the set of indices such that $\mathfrak{u}[i] = 1$. The KGC then randomly selects $s \in_R \mathbb{Z}_p$, and sets $g_2 = (U^*)^s$, where $U^* = u' \prod_{i \in \mathcal{U}} \hat{u}_i$. Other parameters in the master public/private key pair are generated normally by the KGC. In the challenge phase of **Game-II**, the KGC submits two distinct equal-length messages, (m_0, m_1), and ID^* as the target identity. The challenger $\mathcal{C}_2$ randomly selects a bit b, computes the challenge ciphertext $C^* = (\hat{C}^*, \mathsf{com}^*, \mathsf{tag}^*)$ according to the encryption algorithm of [17]. Let the challenge ciphertext $\hat{C}^* = (C_1^*, C_2^*, C_3^*, C_4^*)$. According to the specification of the encryption algorithm in [17], we have

$$C_1^* = e(\mathsf{pk}^{(2)}, g_2)^t M, \quad C_2^* = g^t, \quad C_3^* = (U^*)^t, \quad C_4^* = ((g_1')^{\mathsf{com}^*} h_1)^t$$

where $t \in_R \mathbb{Z}_p$ and $M = m_b \| \mathsf{dec}$ for some binary string dec. The KGC can get the plaintext of $\hat{C}^*$ by computing the following

$$\frac{C_1^*}{e(\mathsf{pk}^{(2)}, (C_3^*)^s)} = \frac{e(\mathsf{pk}^{(2)}, g_2^t) M}{e(\mathsf{pk}^{(2)}, (U^*)^{st})} = \frac{e(\mathsf{pk}^{(2)}, g_2^t) M}{e(\mathsf{pk}^{(2)}, (g_2)^t)} = M = m_b \| \mathsf{dec}$$

By comparing m_b with m_0 and m_1, the KGC can easily find out the message corresponding to C^*. This attack causes the same extent of damage as that described in [2] against [1,14,15]. Both attacks require the KGC to *pre-select* a target identity. The KGC is not able to compromise two users in the system under the **Game-II**. Specific to this certificateless encryption scheme described in [17], there is a more powerful malicious-but-passive KGC attacking technique which allows the KGC to decrypt any ciphertext in the system regardless which user is the corresponding decryptor. Therefore, the KGC does not need to pre-select a target identity.

Attack 2. Note that the message M is 'masked' in C_1^* by $e(\mathsf{pk}^{(2)}, g_2)^t$. Instead of selecting g_2 randomly from $\mathbb{G}_1$, suppose the malicious-but-passive KGC randomly picks $\beta \in \mathbb{Z}_p$ and sets $g_2 = g^\beta$. We can see that the KGC can remove the mask of any ciphertext by simply computing the mask value as $e(\mathsf{pk}^{(2)}, C_2^*)^\beta$.

The certificateless signature scheme of [17] also suffers from the first malicious-but-passive KGC attack described above. We also remark that our attacks do not refute the security claims made in [17] as their security model does not consider/capture the malicious-but-passive KGC attacks.

***A Design Philosophy*:** To design a certificateless scheme, no matter it is an encryption scheme or a signature scheme, for security against malicious-but-passive KGC attacks, we have a great concern on its security if the scheme requires the user to make use of the parameters generated by the KGC when generating its own user public/private key pair (via UserKeyGen). The attacks above illustrate how subtle an attack can be if the malicious-but-passive KGC has certain control on the parameters used for user key pair generation. In the first attack above, the KGC can simply modify the generation of g_2 which is one of the many parameters generated/controlled by the KGC. The user has no way to tell if g_2 is generated accordingly or maliciously. In fact, there are many other ways for the KGC to break the scheme in [17], as there are many parameters generated by or under control of the KGC. One can easily come up with more attacks against the one in [17] in addition to the two attacks described above. In addition, even if the KGC only generates a set of group parameters, for example, a bilinear pairing operation and its associated groups, we cannot guarantee that any malicious-but-passive KGC attack cannot be launched. The reason is that the groups generated and the bilinear operation chosen by the malicious KGC may not be 'generic' [19]. There may exist some trapdoor such that only the one who generates the group parameters, in our case, it is the malicious KGC, can perform some operations efficiently. Therefore, those schemes which require the user to use group parameters generated by the KGC may either be broken by some newly discovered malicious-but-passive KGC attacking techniques, or have their security left unproven.

In our generic scheme proposed below, we design our scheme such that the user partial key and user secret key are generated and used totally independently, while retaining high efficiency.

4 Our Scheme

In this section, we propose a generic certificateless encryption scheme CLE and show that it is secure under the adversarial model defined in Sec. 2. In particular, this generic scheme is the first one proven secure against the malicious-but-passive KGC attacks in the standard model.

4.1 The Scheme

Let $\mathsf{IBE} = (\mathsf{KG}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-ID-CCA2 secure identity-based encryption scheme, $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ an IND-CCA2 secure public key encryption scheme, and $\mathsf{S} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$ a strong one-time signature scheme. Due to page limitation, we skip the formal definitions of them. In the following, we propose a generic certificateless encryption scheme CLE, which is based on these three primitives.

- MasterKeyGen: The KGC runs $(mpk, msk) \leftarrow \mathsf{IBE.KG}(1^k)$, publishes mpk and keeps msk secret.
- PartialKeyGen: On input an identity ID, the KGC runs $psk_{\mathsf{ID}} \leftarrow \mathsf{IBE.Extract}(msk, \mathsf{ID})$ and returns psk_{ID}.
- UserKeyGen: The user (with identity ID) runs $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{PKE.KG}(1^k)$, publishes $(\mathsf{ID}, upk_{\mathsf{ID}})$ and stores usk_{ID}.

- Enc: To encrypt a message m for user ID, the encryptor computes the following and returns $c \overset{\text{def}}{=} (c_2, \sigma, vk)$:

$$\begin{aligned}(vk, sk) &\leftarrow \mathsf{S.KG}(1^k)\\ c_1 &\leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}, m\|vk)\\ c_2 &\leftarrow \mathsf{PKE.Enc}(upk_{\mathsf{ID}}, c_1)\\ \sigma &\leftarrow \mathsf{S.Sign}(sk, c_2)\end{aligned}$$

- Dec: On input an identity ID and a ciphertext $c = (c_2, \sigma, vk)$, if $0 \leftarrow \mathsf{S.Vrfy}(vk, \sigma, c_2)$, $\perp$ is returned. Otherwise, the decryptor computes the following:

$$\begin{aligned}c_1 &\leftarrow \mathsf{PK.Dec}(usk_{\mathsf{ID}}, c_2)\\ m\|vk' &\leftarrow \mathsf{IBE.Dec}(psk_{\mathsf{ID}}, \mathsf{ID}, c_1)\end{aligned}$$

  If $vk' \neq vk$, the decryptor outputs $\perp$; otherwise, it outputs m.

Our construction has some similarity with Yum and Lee's [20], both can be considered as sequential combination of public key encryption and identity-based encryption. However, their scheme is insecure under our model. Our construction additionally employs a strong one-time signature which ensures the '*well-formedness*' of a ciphertext and can be proven secure under our model. In [7], Dodis and Katz considered the security of multiple encryption and presented a generic construction of it. They applied the parallel encryption technique, used a freshly generated one-time verification key as the tag for each (tag-based) encryption scheme, and signed the ciphertext using the one-time signing key. Our scheme employs the sequential encryption, and uses a simpler idea. We do not need to consider tags in the encryption schemes.

4.2 Security Analysis

Theorem 1. *The certificateless encryption scheme* CLE *is* Type-I secure*, provided that the underlying identity-based encryption scheme* IBE *is* IND-ID-CCA2 *secure, and the one-time signature scheme* S *is* strongly unforgeable*.*

Proof. Suppose that $\mathcal{A}_1$ is a PPT adversary that tries to break the Type-I security of CLE. Let the challenge ciphertext that $\mathcal{A}_1$ receives be $c^* = (c_2^*, \sigma^*, vk^*)$. We say a ciphertext $c = (c_2, \sigma, vk)$ is valid if σ is a valid signature on c_2 with regard to the verification key vk. We denote by Forge_1 the event that vk^* appears in a decryption query $(\mathsf{ID}, c = (c_2, \sigma, vk^*))$ issued by $\mathcal{A}_1$ such that c is valid and $(c_2, \sigma) \neq (c_2^*, \sigma^*)$. Then we have:

Lemma 1. $\Pr[\mathsf{Forge}_1]$ *is* negligible *in the security parameter k.*

Lemma 2. $\left|\Pr[\mathcal{A}_1\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] + \frac{1}{2}\Pr[\mathsf{Forge}_1] - \frac{1}{2}\right|$ *is* negligible *(in k).*

Intuitively, since S is a strong one-time signature scheme, a PPT adversary cannot produce a valid forgery using the verification key in the challenge ciphertext with non-negligible probability, so Lemma 1 holds. On the other hand, if an adversary breaks the Type-I security of CLE, we can easily derive an algorithm to break the $\mathsf{IND\text{-}ID\text{-}CCA2}$ security of IBE. Due to page limitation, we omit the proofs here and will present them in the full paper. Therefore, by these two lemmas, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}_1} &= \left|\Pr[\mathcal{A}_1\ \mathsf{Succ}] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathcal{A}_1\ \mathsf{Succ} \wedge \mathsf{Forge}_1] - \frac{1}{2}\Pr[\mathsf{Forge}_1] + \frac{1}{2}\Pr[\mathsf{Forge}_1] + \Pr[\mathcal{A}_1\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] - \frac{1}{2}\right| \\
&\leq \left|\Pr[\mathcal{A}_1\ \mathsf{Succ} \wedge \mathsf{Forge}_1] - \frac{1}{2}\Pr[\mathsf{Forge}_1]\right| + \left|\Pr[\mathcal{A}_1\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] + \Pr[\mathsf{Forge}_1]\cdot\frac{1}{2} - \frac{1}{2}\right| \\
&\leq \frac{1}{2}\Pr[\mathsf{Forge}_1] + \left|\Pr[\mathcal{A}_1\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] + \frac{1}{2}\Pr[\mathsf{Forge}_1] - \frac{1}{2}\right|
\end{aligned}$$

which is also *negligible* in k. This completes the proof of Theorem 1. □

Theorem 2. *The certificateless encryption scheme* CLE *is* Type-II secure *if the underlying public key encryption scheme* PKE *is* $\mathsf{IND\text{-}CCA2}$ *secure and the one-time signature scheme* S *is* strongly unforgeable.

Proof. Suppose that $\mathcal{A}_2$ is a PPT adversary that tries to break the Type-II security of CLE. Let $c^* = (c_2^*, \sigma^*, vk^*)$ be the challenge ciphertext that $\mathcal{A}_2$ receives, and let Forge_2 be the event that vk^* appears in a decryption query $(\mathsf{ID}, c = (c_2, \sigma, vk))$ issued by $\mathcal{A}_2$ (i.e., $vk^* = vk$) such that c is valid and $(c_2, \sigma) \neq (c_2^*, \sigma^*)$. We have the following two lemmas:

Lemma 3. $\Pr[\mathsf{Forge}_2]$ *is* negligible *in* k*.*

This proof is similar to that of Lemma 1, and thus is omitted here.

Lemma 4. $\left|\Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}\Pr[\mathsf{Forge}_2] - \frac{1}{2}\right|$ *is* negligible *in* k*.*

Proof. We construct a PPT algorithm $\mathcal{C}_2$ to break the $\mathsf{IND\text{-}CCA2}$ security of PKE by using $\mathcal{A}_2$ as a subroutine. Given the challenge public key pk and a decryption oracle $\bar{\mathcal{O}}_{\mathsf{D}}$, $\mathcal{C}_2$ runs $\mathcal{A}_2$ on input 1^k, which returns a master public key mpk. Assume that $\mathcal{A}_2$ issues at most q distinct $\mathsf{CreateUser}$ queries. Then, $\mathcal{C}_2$ randomly selects $i \in \{1, 2, \cdots, q\}$, runs $(vk^*, sk^*) \leftarrow \mathsf{S.KG}(1^k)$, stores (vk^*, sk^*) which will be used in the generation of the challenge ciphertext of $\mathcal{A}_2$, and simulates all the oracles for $\mathcal{A}_2$ as follows:

- $\mathsf{CreateUser}$: On input an identity ID and the user partial key psk_{ID}, we assume that this is the j-th distinct $\mathsf{CreateUser}$ query. If $j \neq i$, $\mathcal{C}_2$ runs $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{PKE.KG}(1^k)$, stores $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}}, 0)$ into a list List, and returns upk_{ID}. If $j = i$, $\mathcal{C}_2$ simply stores $(\mathsf{ID}, psk_{\mathsf{ID}}, pk, \perp, 0)$ into List and returns pk.

- $\mathsf{RevealSecretKey}$: On input an identity ID, $\mathcal{C}_2$ searches List for the entry of ID. If there is no such an entry, $\mathcal{C}_2$ returns $\perp$. If $upk_{\mathsf{ID}} = pk$, $\mathcal{C}_2$ outputs a random bit and aborts. Otherwise, it returns the corresponding usk_{ID}.
- $\mathsf{ReplaceKey}$: On input $(\mathsf{ID}, upk', usk')$, $\mathcal{C}_2$ searches List for the entry of ID. If there is no such an entry, $\mathcal{C}_2$ does nothing. If $upk_{\mathsf{ID}} = pk$, $\mathcal{C}_2$ aborts and outputs a random bit. If $usk' = \perp$, $\mathcal{C}_2$ sets $usk' = usk_{\mathsf{ID}}$. Then, it updates $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ to be $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$.
- $\mathsf{Decryption}$: On input $(\mathsf{ID}, c = (c_2, \sigma, vk))$, $\mathcal{C}_2$ searches List for the entry of ID. If it is not found, $\mathcal{C}_2$ returns $\perp$. If ID is not the i-th distinct query made by $\mathcal{A}_2$ to $\mathsf{CreateUser}$, $\mathcal{C}_2$ simulates the $\mathsf{Decryption}$ oracle using its knowledge of psk_{ID} and usk_{ID}. Otherwise ($upk_{\mathsf{ID}} = pk$), after the validity check of σ on c_2 with respect to vk, $\mathcal{C}_2$ makes a $\mathsf{Decryption}$ query to oracle $\bar{\mathcal{O}}_{\mathsf{D}}$ which returns with c_1. Then it completes the rest using its knowledge of psk_{ID}. Note that if event Forge_2 occurs during the simulation of $\mathsf{Decryption}$ oracle, namely, σ is a valid signature on c_2 with respect to vk and $vk = vk^*$, $\mathcal{C}_2$ outputs a random bit and aborts.

At some point $\mathcal{A}_2$ submits two equal-length messages (m_0, m_1) along with a target identity ID^*. If $usk_{\mathsf{ID}^*} \neq pk$, $\mathcal{C}_2$ outputs a random bit and aborts. Otherwise, it computes $(c_1)_0 \leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}^*, m_0 \| vk^*)$ and $(c_1)_1 \leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}^*, m_1 \| vk^*)$, submits $(c_1)_0$ and $(c_1)_1$ to its challenger, and is returned c_2^* which is a ciphertext of $(c_1)_b$, where $b \in \{0,1\}$. It then computes $\sigma^* \leftarrow \mathsf{S.Sign}(sk^*, c_2^*)$ and returns $c^* = (c_2^*, \sigma^*, vk^*)$ to $\mathcal{A}_2$ as the challenge ciphertext. $\mathcal{C}_2$ continues to simulate all the oracles for $\mathcal{A}_2$ as above. Finally it outputs the bit b' output by $\mathcal{A}_2$.

Obviously, the probability that $\mathcal{C}_2$ doesn't abort in simulating $\mathsf{ReplaceKey}$ and $\mathsf{RevealSecretKey}$ oracles is at least $1/q$. If $\mathcal{C}_2$ doesn't abort in simulating the two oracles, the probability that it wins its own game is $\frac{1}{2}\Pr[\mathsf{Forge}_2] + \Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}]$. Thus, we get that the advantage that $\mathcal{C}_2$ wins its game is

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{C}_2} &= \left|\Pr[\mathcal{C}_2\ \mathsf{Succ}] - \frac{1}{2}\right| \\
&\geq \left|\frac{1}{2}\cdot\left(1 - \frac{1}{q}\right) + \left(\Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}\Pr[\mathsf{Forge}_2]\right)\cdot\frac{1}{q} - \frac{1}{2}\right| \\
&= \left|\Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}\Pr[\mathsf{Forge}_2] - \frac{1}{2}\right| \cdot \frac{1}{q}
\end{aligned}$$

Guaranteed by the $\mathsf{IND\text{-}CCA2}$ security of PKE, we have that $\mathsf{Adv}_{\mathcal{C}_2}$ is negligible in k. Thus, $\left|\Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}\Pr[\mathsf{Forge}_2] - \frac{1}{2}\right|$ is negligible as well since $1/q$ is polynomial in k. □

By the two lemmas above, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}_2} &= \left|\Pr[\mathcal{A}_2\ \mathsf{Succ}] - \frac{1}{2}\right| \\
&\leq \left|\Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \mathsf{Forge}_2] - \frac{1}{2}\Pr[\mathsf{Forge}_2]\right| + \left|\Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}\Pr[\mathsf{Forge}_2] - \frac{1}{2}\right| \\
&\leq \frac{1}{2}\Pr[\mathsf{Forge}_2] + \left|\Pr[\mathcal{A}_2\ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}\Pr[\mathsf{Forge}_2] - \frac{1}{2}\right|
\end{aligned}$$

which is negligible as well. This completes the proof of Theorem 2. □

The following corollary is obtained immediately from Theorem 1 and Theorem 2.

Corollary 1. *The certificateless encryption scheme* CLE *described above is* IND-ID-CCA2 secure*.*

4.3 Efficiency

In practice, for high performance in the encryption process, we usually use the *hybrid encryption* method which combines a public key encryption and a symmetric encryption to encrypt the message instead of using the public key encryption directly to encrypt bulk data. To apply this to our generic scheme CLE, we first generate a random symmetric key key for a secure symmetric encryption scheme SE, then use SE to encrypt $m\|vk$ under key, and finally encrypt key using CLE. The decryption algorithm is modified accordingly. One of the key advantages of applying the hybrid encryption onto CLE (besides efficiency) is that the message space will not be restricted by the size of the verification key vk. We elaborate more on the size of vk below.

The (strong) one-time signature in our generic scheme CLE provides a certain assurance on that the encryptor did encrypt the message itself (or the '*well-formedness*' of a ciphertext). Since most of the one-time signature schemes in the literature follow the 'one-way function paradigm' [9,11], the verification key and the signature are both of large size. An immediate consequence is that the message encrypted (i.e., $m\|vk$) and the resulting ciphertext (i.e., $c = (c_2, \sigma, vk)$) of our scheme also suffer from the large size. We describe two methods which can significantly reduce the size in the actual implementation of CLE.

One simple solution is to replace the strong one-time signature scheme with a conventional signature scheme which is strongly unforgeable under adaptive chosen message attack. This does not weaken the security of CLE because any strongly unforgeable signature scheme is also a strong one-time signature scheme. A good candidate is the short signature proposed by Boneh and Boyen [3] as the verification key and the signature of the scheme in [3] are both small in size. The tradeoff is that the resulting scheme requires more computation than that of a strong one-time signature scheme.

Another solution is to first map the one-time verification key into a much shorter string using a collision-resistant hash function H, and then encrypt $m\|\mathsf{H}(vk)$ rather than $m\|vk$. The decryption algorithm is changed accordingly. That is, the user checks if the second part of the plaintext (decrypted from the ciphertext) is the hash of vk. To analyze the security of this modified scheme, we need to show that it is negligible for the adversary to issue a $\mathsf{Decryption}$ query on input ID and $c = (c_2, \sigma, vk)$ such that $vk \neq vk^*$ but $\mathsf{H}(vk) = \mathsf{H}(vk^*)$, where vk^* is the verification key in the challenge ciphertext. It is guaranteed by the collision-resistance property of H. This method reduces the impact on the message size by the size of the verification key while adding only slightly on the computation cost. However, the ciphertext is not much shorter than that of the original scheme.

We also observe that the verification key vk could be removed from the ciphertext without any influence on the security of the resulting encryption scheme. In addition, there is no need to use the asymmetric version of one-time signature. Instead, we can use the symmetric version - a (strong) one-time message authentication code (MAC) [5,4]. The construction of the new scheme CLE' remains the same as CLE, except that the one-time signature is replaced with a MAC. It is readily to see that CLE' enjoys better efficiency and much shorter ciphertext than CLE does.

5 Conclusion

In this paper, we considered the security of certificateless encryption schemes in the presence of a *malicious-but-passive* KGC, and proposed the *first generic* certificateless encryption scheme in *the standard model*. We also described how to use short signature or hybrid encryption to implement efficient instantiations of our generic scheme.

On the study of malicious-but-passive KGC attacks, we showed that although the scheme in [17] does not have the same key generation procedure as that of [1], there are two new attacks which can compromise the Type-II security of their scheme. In particular, our second attack allows the KGC to decrypt any ciphertext without pre-selecting a target user.

It still remains an open problem to determine the exact relationship between strong decryption oracle introduced in [1] and the malicious-but-passive KGC adversarial capability.

Acknowledgements

We would like to thank all the anonymous reviewers of IWSEC 2007 for their invaluable comments.

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Au, M.H., Chen, J., Liu, J.K., Mu, Y., Wong, D.S., Yang, G.: Malicious KGC attacks in certificateless cryptography. In: ACM ASIACCS 2007, pp. 302–311. ACM Press, New York (2007), http://eprint.iacr.org/2006/255
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Computing 36(5), 915–942 (2006)

5. Boneh, D., Katz, J.: Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
6. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
7. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)
8. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM J. Computing 30(2), 391–437 (2000)
9. Even, S., Goldreich, O., Micali, S.: On-line/off-line digital signatures. J. of Cryptology 9(1), 35–67 (1996)
10. Galindo, D., Morillo, P., Rafols, C.: Breaking Yum and Lee generic constructions of certificate-less and certificate-based encryption schemes. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 81–91. Springer, Heidelberg (2006)
11. Goldreich, O.: Foundations of Cryptography, Basic Applications, vol. II. Cambridge University Press, Cambridge (2004)
12. Hu, B.C., Wong, D.S., Zhang, Z., Deng, X.: Key replacement attack against a generic construction of certificateless signature. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 235–246. Springer, Heidelberg (2006)
13. Hu, B.C., Wong, D.S., Zhang, Z., Deng, X.: Certificateless signature: A new security model and an improved generic construction. Designs, Codes, and Cryptography 42(2), 109–126 (2007)
14. Huang, X., Susilo, W., Mu, Y., Zhang, F.: On the security of certificateless signature schemes from Asiacrypt 2003. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 13–25. Springer, Heidelberg (2005)
15. Li, X., Chen, K., Sun, L.: Certificateless signature and proxy signature schemes from bilinear pairings. Lithuanian Mathematical Journal 45(1), 76–83 (2005)
16. Libert, B., Quisquater, J.-J.: On constructing certificateless cryptosystems from identity based encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006)
17. Liu, J.K., Au, M.H., Susilo, W.: Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In: ACM ASIACCS'07, pp. 273–283. ACM Press, New York (2007), Full paper at http://eprint.iacr.org/2006/373
18. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
19. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
20. Yum, D.H., Lee, P.J.: Generic construction of certificateless encryption. In: Laganà, A., Gavrilova, M., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. LNCS, vol. 3043, pp. 802–811. Springer, Heidelberg (2004)
21. Yum, D.H., Lee, P.J.: Generic construction of certificateless signature. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 200–211. Springer, Heidelberg (2004)

On Security Models and Compilers for Group Key Exchange Protocols

(Extended Abstract*)

Emmanuel Bresson[1], Mark Manulis[2,**], and Jörg Schwenk[2]

[1] DCSSI Crypto Lab Paris
emmanuel@bresson.org
[2] Horst Görtz Institute, Ruhr University Bochum, Germany
{mark.manulis, joerg.schwenk}@nds.rub.de

Abstract. Group key exchange (GKE) protocols can be used to guarantee confidentiality and authentication in group applications. The paradigm of provable security subsumes an abstract formalization (security model) that considers the protocol environment and identifies its security goals. The first security model for GKE protocols was proposed by Bresson, Chevassut, Pointcheval, and Quisquater in 2001, and has been subsequently applied in many security proofs. Their definitions of AKE-security (authenticated key exchange; a.k.a. indistinguishability of the key) and MA-security (mutual authentication) became meanwhile standard.

In this paper we analyze the BCPQ model and some of its variants and identify several risks resulting from its technical core construction – the notion of *partnering*. Consequently, we propose a revised model extending AKE- and MA-security in order to capture attacks by malicious participants and strong corruptions.

Then, we turn to generic solutions (known as *compilers*) for AKE- and MA-security in BCPQ-like models. We describe a compiler C-AMA which provides AKE- and MA-security for any GKE protocol, under standard cryptographic assumptions, that eliminates some identified limitations in existing compilers.

1 Introduction

MOTIVATION. Security of many privacy-preserving multi-party applications like encrypted group communication for audio/video conferences, chat systems, computer-supported collaborative workflow systems, or secure server replication systems depends on group key exchange (GKE) protocols. Security of those latter is therefore critical. Security of earlier GKE protocols [29, 41, 17, 42, 2, 38, 33, 34] has been analyzed heuristically based on informal definitions so that some of them have been broken later, e.g., [37]. By contrast, the paradigm of *provable security* is used across the modern literature to prove in a mathematical way, and under reasonable assumptions, that a cryptographic scheme achieves the required security goals. Such a proof usually uses a formal setting that specifies: (1) the computing environment (involved users,

[*] A full version of this work is available at http://eprint.iacr.org/2006/385/

[**] The author was supported by the European Commission (IST-2002-507932 ECRYPT).

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 292–307, 2007.
© Springer-Verlag Berlin Heidelberg 2007

their trust relationship, cryptographic parameters, communication...), (2) the adversarial environment and (3) the definitions of some concrete security goals. In 2001 Bresson, Chevassut, Pointcheval, and Quisquater [14] introduced the first security model (BCPQ model) designed for GKE protocols. They adopted some ideas previously proposed by Bellare and Rogaway [5,7] in the context of two- and three-party key establishment protocols. The BCPQ model, as well as its refinements [11,12] and variants [31,13,27,30], have been meanwhile used in many GKE security proofs including [14,11,12,13,31,10,32,27,26,30,1] and became *de facto* standard.

MODULARITY OF PROTOCOL DESIGN. Seeing GKE protocols as building blocks for high-level applications, it is worth designing protocols that have specific security goals. Modular design allows to build such "*à la carte*" protocols. In order to provide these modular constructions in a generic way, so-called "compilers" have been developed, e.g., [31,30]. They allow designers to enhance security of a protocol in a *black-box* manner, that is, independently of the implementation of the protocol being enhanced.

CONTRIBUTIONS AND ORGANIZATION. We start with a brief overview of the BCPQ; in Section 2.1 we point out a problem in the BCPQ model between its technical core – the notion of *partnering* – and its definition of MA-security. Next in Section 2.2 we analyze some variants of the BCPQ model, from the perspective of MA-security, in particular in presence of *malicious participants*. By malicious participants we mean legitimate protocol participants who are fully controlled by the adversary. We emphasize that consideration of malicious participants makes sense in the scope of MA-security but not of AKE-security.

After identifying some drawbacks in the mentioned variants we describe in Section 3 an extended security model for revised definitions of AKE- and MA-security with consideration of malicious participants. Our model is based on the more powerful BCPQ refinement from [12] that considers AKE-security in the presence of (partial) internal state corruptions (strong corruptions). We also introduce an additional notion of *backward secrecy* which leads to new corruption models in case of AKE-security.

In Section 4.2 we provide a brief analysis of some known security-enhancing compilers for GKE protocols. In particular, we show that the compiler proposed by Katz and Yung in [31] needs some additional assumptions in order to be considered as a really generic solution. To overcome this, and in order to show that our extended security definitions are feasible enough for the construction of practical reductionist security proofs, we describe in Section 4.3 a compiler C-AMA that satisfies our *stronger* definitions of AKE- and MA-security. We further prove its security under standard cryptographic assumptions.

2 The BCPQ Model and Its Variants

The BCPQ model extends the methodology introduced by Bellare and Rogaway [6,7] to a group setting. The model considers a set ID of n participants. Each $U_i \in ID$ has a long-lived key LL_i and an unlimited number of instances called *oracles* and denoted $\Pi_i^{s_i}$ (s_i-th instance of U_i) involved in different executions of P. The BCPQ model uses session ids to define a *"partnering"* relation which is used to define security goals.

The session id of an oracle Π_i^s is defined as $\mathrm{SID}(\Pi_i^s) := \{\mathrm{SID}_{ij} \mid U_j \in ID\}$ where SID_{ij} is the concatenation of all flows that Π_i^s exchanges with any oracle Π_j^t. Then two oracles Π and Π' are called *directly partnered*, denoted $\Pi \leftrightarrow \Pi'$, if both oracles *accept* (compute the session key) and if $\mathrm{SID}(\Pi) \cap \mathrm{SID}(\Pi') \neq \emptyset$. Further, oracles Π and Π' are *partnered*, denoted $\Pi \leftrightsquigarrow \Pi'$, if, in the graph $G_{\mathrm{SIDS}} := (V, E)$ with $V := \{\Pi_l^s | \, l \in [1, n], s \in \mathbb{N}\}$ and $E := \{(\Pi_i, \Pi_j) | \; \Pi_i \leftrightarrow \Pi_j\}$, there exists a sequence of oracles $(\Pi_1, \Pi_2, \dots, \Pi_k)$ with $\Pi = \Pi_1$, $\Pi' = \Pi_k$, and $\Pi_{l-1} \leftrightarrow \Pi_l$ for all $l \in [2, k]$. The *partner id* for an oracle Π_i^s is $\mathrm{PIDS}(\Pi_i^s) = \{\Pi_l^t \mid \Pi_i^s \leftrightsquigarrow \Pi_l^t \; \forall \, l \in [1, n] \setminus \{i\}\}$.

The BCPQ model considers a Probabilistic Polynomial-Time (PPT) active adversary $\mathcal{A}$ which can send messages to the oracles via a Send query, reveal the session key via a Reveal query, obtain a long-lived key via a Corrupt query, and ask a Test query to obtain either a session key or a random number. Using this adversarial setting two security goals are specified for a GKE protocol: AKE-security and MA-security.

The AKE-security requires that adversary $\mathcal{A}$ asks a single Test query to a fresh oracle. An oracle $\Pi_i^{s_i}$ is *fresh* if (1) it has accepted, (2) no oracle has been asked for a Corrupt query before $\Pi_i^{s_i}$ accepts, and (3) neither $\Pi_i^{s_i}$ nor any of its partners have been asked for a Reveal query. A GKE protocol is said to be AKE-secure if $\mathcal{A}$ cannot guess which value it has received in response to its Test query, i.e., the session key or a random number, significantly better than at random. This definition of AKE-security encompass earlier informal definitions:

- *key secrecy* [24] (a.k.a. *implicit key authentication* [36]) which ensures that no other party except for legitimate participants learns the established group key;
- *resistance against known-key attacks* [43,16] (a.k.a. *key independence* [33]) meaning that an adversary knowing group keys of other sessions cannot compute subsequent session keys;
- *perfect forward secrecy* [28,24,36] requiring that the disclosure of long-term keying material must not compromise the secrecy of the established keys from earlier protocol runs.

The MA-security means that $\mathcal{A}$ cannot impersonate a participant U_i through its oracle $\Pi_i^{s_i}$. Impersonation would imply that there exists one oracle $\Pi_i^{s_i}$ which accepts with $|\mathrm{PIDS}(\Pi_i^{s_i})| \neq n-1$ must be negligible. In other words, if each oracle $\Pi_i^{s_i}$ accepts with $|\mathrm{PIDS}(\Pi_i^{s_i})| = n-1$ then no impersonation attacks could have occurred — thus the informal notion of *mutual authentication* [6] is satisfied. Further, we recall the following claims given by the authors of [14]:

> "In the definition of partnering, we do not require that the session key computed by partnered oracles be the same since it can easily be proven that the probability that **partnered** oracles come up with different session keys is negligible." [14, Footnote 3]

> "We are not concerned with partnered oracles coming up with different session keys, since our definition of partnering implies the oracles have exchanged *exactly* the same flows." [14, Section 7.4]

If these claims hold then MA-security captures the following informal earlier security goals:

- *key confirmation*: all participants that have accepted[1] hold identical group keys;
- *explicit key authentication* [36]: both key confirmation and mutual authentication.

2.1 Problems with the Definition of MA-Security in the BCPQ Model

In this section, we explain why the definition of MA-security might not be general enough for GKE protocols. We do not pretend having broken some provably MA-secure scheme. In contrast, we explain why, if every participating oracle $\Pi_i^{s_i}$ accepts with $|\mathrm{PIDS}(\Pi_i^{s_i})| = n-1$, it does not necessarily mean that the considered protocol provides mutual authentication and key confirmation:

1. There exist GKE protocols where an active adversary $\mathcal{A}$ can impersonate one of the participants through its oracle but nevertheless every participating oracle $\Pi_i^{s_i}$ accepts with $|\mathrm{PIDS}(\Pi_i^{s_i})| = n-1$.
2. There exist GKE protocols where each oracle $\Pi_i^{s_i}$ accepts with $|\mathrm{PIDS}(\Pi_i^{s_i})| = n-1$ but there are at least two partnered oracles that have computed different keys.

We start from the protocols presented in [14, 11, 12, 13]. We first study the case at an abstract level. On Figure 1, we show an execution with 3 participants in which $\mathcal{A}$ impersonates U_1 and modifies message m_1 to $\tilde{m}_1$ (Figure 1) such that $\mathrm{SID}_{21} = \tilde{m}_1$ (Figure 2 for the construction of sessions ids).

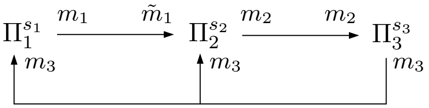


Fig. 1. Protocol execution where $\mathcal{A}$ impersonates U_1

$\mathrm{SID}(\Pi_i^{s_i})$	SID_{i1}	SID_{i2}	SID_{i3}
$\mathrm{SID}(\Pi_1^{s_1})$	$\emptyset$	m_1	m_3
$\mathrm{SID}(\Pi_2^{s_2})$	$\tilde{m}_1$	$\emptyset$	$m_2 \vert m_3$
$\mathrm{SID}(\Pi_3^{s_3})$	m_3	$m_2 \vert m_3$	$\emptyset$

Fig. 2. $\mathrm{SID}(\Pi_i^{s_i})$ in the protocol execution with impersonation of U_1

We may assume that there exists protocols for which all oracles accept after this modification. Here every oracle $\Pi_i^{s_i}$ accepts with $|\mathrm{PIDS}(\Pi_i^{s_i})| = 2$. To check this we first note that $\mathrm{SID}_{12} = m_1$. Though $\mathrm{SID}(\Pi_1^{s_1}) \cap \mathrm{SID}(\Pi_2^{s_2}) = \{m_1, m_3\} \cap \{\tilde{m}_1, m_2|m_3\} = \emptyset$ and thus $\Pi_1^{s_1} \not\leftrightsquigarrow \Pi_2^{s_2}$, we still have $\mathrm{SID}(\Pi_1^{s_1}) \cap \mathrm{SID}(\Pi_3^{s_3}) = \{m_1, m_3\} \cap \{m_3, m_2|m_3\} = m_3$ and $\mathrm{SID}(\Pi_3^{s_3}) \cap \mathrm{SID}(\Pi_2^{s_2}) = \{m_3, m_2|m_3\} \cap \{\tilde{m}_1, m_2|m_3\} = m_2|m_3$ so that $\Pi_1^{s_1} \leftrightsquigarrow \Pi_2^{s_2}$: all oracles are still partnered despite the impersonation attack. However, oracle $\Pi_2^{s_2}$ has received a modified message and this may result in different keys computed by $\Pi_1^{s_1}$ and $\Pi_2^{s_2}$.

CONCRETE EXAMPLE. As a concrete example, we illustrate how a replay attack would work on the protocol from [14] if the additional confirmation round is missing. On Figure 3, $[m]_{U_i}$ denotes a message m signed by $\Pi_i^{s_i}$, and $V(m) \stackrel{?}{=} 1$ its verification; g is a generator of a finite cyclic group.

[1] This is a slightly modified definition from [36] wrt. to the arguments from [39] on impossibility of the assurance of some participant that other participants have actually accepted the key.

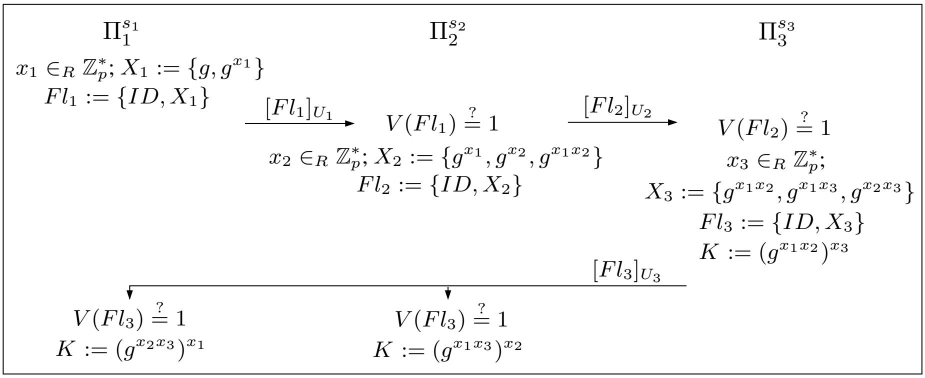

$\Pi_1^{s_1}$
$\Pi_2^{s_2}$
$\Pi_3^{s_3}$
$x_1 \in_R \mathbb{Z}_p^*; X_1 := \{g, g^{x_1}\}$
$Fl_1 := \{ID, X_1\}$
$[Fl_1]_{U_1}$
$V(Fl_1) \overset{?}{=} 1$
$x_2 \in_R \mathbb{Z}_p^*; X_2 := \{g^{x_1}, g^{x_2}, g^{x_1x_2}\}$
$Fl_2 := \{ID, X_2\}$
$[Fl_2]_{U_2}$
$V(Fl_2) \overset{?}{=} 1$
$x_3 \in_R \mathbb{Z}_p^*;$
$X_3 := \{g^{x_1x_2}, g^{x_1x_3}, g^{x_2x_3}\}$
$Fl_3 := \{ID, X_3\}$
$K := (g^{x_1x_2})^{x_3}$
$[Fl_3]_{U_3}$
$V(Fl_3) \overset{?}{=} 1$
$K := (g^{x_2x_3})^{x_1}$
$V(Fl_3) \overset{?}{=} 1$
$K := (g^{x_1x_3})^{x_2}$

Fig. 3. Execution of the protocol in [14] with three participants

Here $\mathcal{A}$ drops the flow $[Fl_1]_{U_1}$ and replays a previously sent message. The replayed message is likely to be $\widetilde{[Fl_1]_{U_1}}$ with $\widetilde{Fl_1} := (ID, \{g, g^{\widetilde{x_1}}\})$ for some $\widetilde{x_1} \neq x_1$. Obviously, $V(\widetilde{Fl_1}) = 1$ holds. It is easy to see that $X_2 = \{g^{\widetilde{x_1}}, g^{x_2}, g^{\widetilde{x_1}x_2}\}$ and $X_3 := \{g^{\widetilde{x_1}x_2}, g^{\widetilde{x_1}x_3}, g^{x_2x_3}\}$ so that $\Pi_1^{s_1}$ computes $K = g^{x_1x_2x_3}$ whereas $\Pi_2^{s_2}$ and $\Pi_3^{s_3}$ compute another value, i.e., $K = g^{\widetilde{x_1}x_2x_3}$. Moreover, it is easy to check that $|\mathrm{PIDS}(\Pi_i^{s_i})| = 2$ for every $\Pi_i^{s_i}$, $i \in \{1, 2, 3\}$.

Thus $|\mathrm{PIDS}(\Pi_i^{s_i})| = n-1$ does not ensure mutual authentication and key confirmation. We stress that this does not contradict the MA-security of the proposed protocol when the additional round is executed. However, it is worth studying the notion of MA-security independently from some concrete protocol design.

Furthermore, we stress that a more general definition of MA-security should also consider attacks by malicious protocol participants (AKE-security in such setting is hopeless). As noted in [21], this is the reason the BCPQ model fails to provide security against *unknown key-share attacks* [9]. The same remark is done in [20] for the protocol from [10].

2.2 MA-Security in Some BCPQ-Variants

VARIANTS BY KATZ-YUNG AND DUTTA *et al*. These are two modifications [31,27] to the BCPQ model, that propose a different construction of partnering. However neither of them propose a specific definition of MA-security, and they just refer to BCPQ.

A VARIANT BY KATZ AND SHIN. Katz and Shin [30] proposed a different security model (KS) for GKE protocols, and provide a security analysis in the framework of Universal Composability (UC) [18]. This provides the first formal treatment of GKE protocols security in the presence of malicious participants. Here unique session ids are assumed to be provided by some high-level application mechanism [3], and partner ids is just the set of users whose oracle intend to establish a key. In addition to their model, Katz and Shin proposed a compiler to turn any GKE protocol which is secure in the BCPQ model into a protocol which is secure in the KS model.

Among other things, the KS model defines *security against insider attacks* as a combination of two requirements: *agreement* and *security against insider impersonation attacks*:

- the adversary $\mathcal{A}$ *violates agreement* if two partnered, uncorrupted oracles accept with different session keys.
- the adversary $\mathcal{A}$ *impersonates uncorrupted* U_j *to accepting* Π_i^s if U_j belongs to the (expected) partner id of Π_i^s but in fact no oracle Π_j^t is partnered with Π_i^s. In other words, the instance Π_i^s computes the session key and U_i believes that U_j does so, but in fact an adversary has participated in the protocol on behalf of U_j.

Intuitively, agreement considers key confirmation in case that all other participants are malicious (corrupted); and security against insider impersonation attacks considers mutual authentication and unknown key-share resilience in the presence of malicious participants.

EXTENSIONS BY BOHLI *et al.* AND DESMEDT *et al.* These extensions of the BCPQ model takes consideration of attacks whose goal is to "bias" the value of the group key: Bohli *et al.* [8] consider malicious participants who *deliberately* wish to influence the key. Desmedt *et al.* [22] formalized a related notion called *non-malleability*. In our parallel work [15] we propose a generic compiler C-MACON which addresses these security requirements from the perspective of malicious participants.

3 The Revised "Game-Based" Security Model

In the following we extend the BCPQ model under consideration of malicious participants. We revisit the definition of MA-security, so that the unified definition can replace both definitions of agreement and security against insider impersonation attacks of the KS model.

Our refinements are in two directions: *strong corruptions* (in the sense of [12]), and a new requirement which we call *backward secrecy*.

3.1 Protocol Participants, Variables

USERS, INSTANCE ORACLES. We consider $\mathcal{U}$ as a set of N users in the universe. Each user $U_i \in \mathcal{U}$ holds a long-lived key LL_i. In order to handle participation of U_i in distinct concurrent protocol executions we consider that U_i has an unlimited number of instances called *oracles*; Π_i^s, with $s \in \mathbb{N}$, denotes the s-th instance oracle of U_i.

INTERNAL STATES. Every Π_U^s maintains an *internal state information* $\mathtt{state}_U^s$ which is composed of all private, ephemeral information used during the protocol execution. The long-lived key LL_U is, in nature, excluded from it (moreover the long-lived key is specific to the user, not to the oracle).

SESSION GROUP KEY, SESSION ID, PARTNER ID. In each session we consider a new group $\mathcal{G}$ of $n \in [1, N]$ participating oracles. Each oracle in $\mathcal{G}$ is called a *group member*. Every participating oracle $\Pi_U^s \in \mathcal{G}$ computes the *session group key* $k_U^s \in \{0,1\}^\kappa$.

Every session is identified by a unique *session id* $\mathtt{sid}_U^s$ known to all oracles in the session. Similarly, each oracle $\Pi_U^s \in \mathcal{G}$ gets a value $\mathtt{pid}_U^s$ that contains the identities of participating users (including U). We say that two oracles, $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$, are *partnered* if $U_i \in \mathtt{pid}_j^{s_j}$, $U_j \in \mathtt{pid}_i^{s_i}$, and $\mathtt{sid}_i^{s_i} = \mathtt{sid}_j^{s_j}$.

INSTANCE ORACLE STATES. An oracle Π_U^s is originally *unused* and becomes *used*, initialized with LL_U, when it becomes part of a group $\mathcal{G}$. Then it turns into the *stand-by* state where it waits for an invocation to execute the protocol. Upon invocation, the oracle Π_U^s learns its partner id $\mathtt{pid}_U^s$ (and possibly $\mathtt{sid}_U^s$) and turns into a *processing* state where it sends, receives and processes messages. During this phase, the internal state $\mathtt{state}_U^s$ is maintained by the oracle. When oracle Π_U^s has enough information to compute the session key, it *accepts*. After some possible auxiliary steps, it *terminates* meaning that it would not send or receive further messages. If the execution fails (due to any adversarial actions) then Π_U^s terminates without having accepted, and the key k_U^s is set to some $\mathtt{undefined}$ value.

Definition 1 (GKE Protocol). *A group key exchange protocol* $\mathtt{P}$ *consists of the key generation algorithm* KeyGen*, and a protocol* Setup *defined as follows:*

> $\mathtt{P}$.KeyGen(1^κ)*: On input a security parameter* 1^κ *each user in* $\mathcal{U}$ *is provided with a long-lived key* LL_U*.*
> $\mathtt{P}$.Setup$(\mathcal{S})$*: On input a set* $\mathcal{S}$ *of* n *unused oracles a new group* $\mathcal{G} := \mathcal{S}$ *is created and a probabilistic interactive protocol is executed between oracles in* $\mathcal{G}$*.*

We call $\mathtt{P}$.Setup an *operation*. We say that a protocol is *correct* if all oracles in $\mathcal{G}$ accept with the same group key k and assume it is always the case in this paper.

3.2 Adversarial Model

QUERIES TO THE INSTANCE ORACLES. The adversary $\mathcal{A}$ (passive or active) is represented by a PPT machine and may interact with group members by making the following oracle queries:

- Setup$(\mathcal{S})$: This query models $\mathcal{A}$ eavesdropping an honest operation execution. $\mathtt{P}$.Setup$(\mathcal{S})$ is executed and $\mathcal{A}$ is given the transcript of the execution.
- Send(Π_U^s, m): This query models $\mathcal{A}$ sending messages to the oracles. $\mathcal{A}$ receives the response which Π_U^s would have generated after having processed the message m. When asking Send$(\Pi_U^s, \mathcal{S})$, $\mathcal{A}$ gets the first message of the protocol.
- RevealKey(Π_U^s): $\mathcal{A}$ is given the session group key k_U^s, provided Π_U^s has accepted.
- RevealState(Π_U^s): $\mathcal{A}$ is given the internal state information $\mathtt{state}_U^s$.[2]
- Corrupt(U): $\mathcal{A}$ is given the long-lived key LL_U.
- Test(Π_U^s): This query will be used to model the AKE-security of a GKE protocol. It can be asked at any time, but only once. It is answered as follows: the oracle generates a random bit b. If $b = 1$ then $\mathcal{A}$ is given k_U^s, and if $b = 0$ then $\mathcal{A}$ is given a random string.

In spirit of [19], we restrict a *passive* adversary not to replay, modify or inject messages; but, it can still change their delivery order, drop or delay them.

[2] This kind of the adversarial query has previously been mentioned by Canetti and Krawczyk in their model for two-party protocols [19].

FORWARD SECRECY. A completed session is *"forward-secure"* if its security remains whatever the adversary does in the future. Similar to [12] we distinguish between *weak-forward secrecy* ($\mathtt{wfs}$) where $\mathcal{A}$ is allowed to ask $\mathsf{Corrupt}$ queries, and *strong-forward secrecy* ($\mathtt{sfs}$) where it can also ask $\mathsf{RevealState}$ queries.

BACKWARD SECRECY. The notion of backward secrecy is symmetric to forward secrecy in the sense that it considers damages to the AKE-security of future sessions after actions of the adversary in the past/current sessions. The notion might seem useless at first glance (such actions can make secrecy just impossible), however, there might exist intermediate actions, such as corruptions of internal states, that do not compromise future session keys (or at least not all of them). We distinguish between *weak-backward secrecy* ($\mathtt{wbs}$) where $\mathcal{A}$ is allowed to ask $\mathsf{RevealState}$ queries, and *strong-backward secrecy* ($\mathtt{sbs}$) where it can also ask $\mathsf{Corrupt}$ queries[3].

ORACLE FRESHNESS, CORRUPTION MODELS, ADVERSARIAL SETTINGS. To properly deal with forward and backward secrecy, we need to distinguish cases where the adversary may have participate in the protocol. In order to consider only non-participating adversaries we introduce the following notion of α-fresh sessions.

The notion of freshness for an oracle Π_U^s is needed to distinguish between various definitions of security wrt. different flavors of backward or forward secrecy. Each flavor $\alpha \in \{\emptyset, \mathtt{wbs}, \mathtt{wfs}, \mathtt{sbs}, \mathtt{sfs}\}$ leads to a different definition of freshness.

Definition 2 **(α-Freshness).** *Let* $\alpha \in \{\emptyset, \mathtt{wfs}, \mathtt{wbs}, \mathtt{sfs}, \mathtt{sbs}\}$. *The oracle* $\Pi_U^s \in \mathcal{G}$ *is*

$\emptyset$-fresh: *neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealKey}$ *query after having accepted;*

$\mathtt{wbs}$-fresh: *(1) neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealState}$ *query after* $\mathcal{G}$ *is created, and (2) neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealKey}$ *query after having accepted;*

$\mathtt{wfs}$-fresh: *(1) no* $U_i \in \mathtt{pid}_U^s$ *have been asked for a* $\mathsf{Corrupt}$ *query prior to a query of the form* $\mathsf{Send}(\Pi_j^{s_j}, m)$ *such that* $U_j \in \mathtt{pid}_U^s$ *before* Π_U^s *and all its partners accept, and (2) neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealKey}$ *query after having accepted;*

$\mathtt{sbs}$-fresh: *(1) no* $U_i \in \mathtt{pid}_U^s$ *have been asked for a* $\mathsf{Corrupt}$ *query prior to a query of the form* $\mathsf{Send}(\Pi_j^{s_j}, m)$ *such that* $U_j \in \mathtt{pid}_U^s$ *after* $\mathcal{G}$ *is created, (2) neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealState}$ *query after* $\mathcal{G}$ *is created, and (3) neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealKey}$ *query after having accepted;*

$\mathtt{sfs}$-fresh: *(1) no* $U_i \in \mathtt{pid}_U^s$ *have been asked for a* $\mathsf{Corrupt}$ *query prior to a query of the form* $\mathsf{Send}(\Pi_j^{s_j}, m)$ *such that* $U_j \in \mathtt{pid}_U^s$ *before* Π_U^s *and all its partners accept, (2) neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealState}$ *query before they accept, and (3) neither* Π_U^s *nor any of its partners have been asked for a* $\mathsf{RevealKey}$ *query after having accepted.*

We say that a session is α-fresh *if all participating oracles are* α*-fresh.*

[3] In case of backward secrecy $\mathsf{Corrupt}$ queries are more damageable than $\mathsf{RevealState}$ queries because the long-lived keys are usually used for authentication and their knowledge allows the adversary to impersonate users in subsequent sessions and learn the session group key.

Obviously, the $\mathtt{wfs}$-freshness allows $\mathsf{Corrupt}$ queries to any user *after* the oracles in $\mathcal{G}$ have accepted (and the $\mathtt{sfs}$-freshness allows, additionally, $\mathsf{RevealState}$ queries). Beside this, $\mathsf{Corrupt}$ and $\mathsf{RevealState}$ queries are allowed for oracles outside of $\mathcal{G}$.

The notion of α-fresh sessions becomes important in security proofs in order to distinguish between "honest" and "corrupted" sessions. Intuitively, the adversary will not be allowed to ask some "bad" queries. To properly manage the adversarial capabilities for each scenario of freshness, we distinguish between the following corruption models.

Definition 3 (Corruption Model β). *An adversary $\mathcal{A}$, in addition to* Setup, $\mathsf{RevealKey}$, Send, *and* Test, *may ask* $\mathsf{RevealState}$ *and* $\mathsf{Corrupt}$ *queries as specified by the used corruption model β, which can be one of the following:*

weak corruption model $\mathtt{wcm}$: $\mathcal{A}$ *may ask* neither $\mathsf{RevealState}$ nor $\mathsf{Corrupt}$.
weak corruption model for forward secrecy $\mathtt{wcm\text{-}fs}$: $\mathcal{A}$ *may ask* just $\mathsf{Corrupt}$.
weak corruption model for backward secrecy $\mathtt{wcm\text{-}bs}$: $\mathcal{A}$ *may ask* just $\mathsf{RevealState}$.
strong corruption model $\mathtt{scm}$: $\mathcal{A}$ *may ask* both *queries,* $\mathsf{RevealState}$ *and* $\mathsf{Corrupt}$.

A concrete proof of AKE-security needs to specify capabilities of the adversary depending on the intended freshness type. Combining definitions for freshness and corruptions, we obtain a set of *reasonable adversarial settings* $(\alpha, \beta) \in \{(\emptyset, \mathtt{wcm}), (\mathtt{wfs}, \mathtt{wcm\text{-}fs}), (\mathtt{wbs}, \mathtt{wcm\text{-}bs}), (\mathtt{sbs}, \mathtt{scm}), (\mathtt{sfs}, \mathtt{scm})\}$. Note that other imaginable settings are not reasonable from the perspective of the attacks.

Remark 4. In practice long-lived keys are used to achieve authentication, and thus if an adversary is able to corrupt a group member (obtaining its long-lived key) then it can impersonate that member in subsequent sessions. Therefore, achieving AKE-security in the $(\mathtt{sbs}, \mathtt{scm})$ appears tricky. To the contrary, the adversarial setting $(\mathtt{wbs}, \mathtt{wcm\text{-}bs})$ appears of great interest since it is independent of any long-term secrets. Moreover we argue that $(\mathtt{wbs}, \mathtt{wcm\text{-}bs})$ is important since in previous models [12, 30] a persistent internal state is used in both past and future sessions, and thus, while forward secrecy looks at (state) corruptions in later sessions, backward secrecy must legitimately look at state corruptions in previous sessions.

3.3 Security Goals

In this section we describe security goals for a GKE protocol. We give a formal definition of (Authenticated)KeyExchange-security (indistinguishability of session group keys), and a new definition of MA-security that considers malicious participants and internal state corruptions of honest participants.

Definition 5 ((A)KE-Security). *Let $\mathtt{P}$ be a correct GKE protocol and b a uniformly chosen bit. Consider a reasonable adversarial setting (α, β) and an (active) adversary $\mathcal{A}$. We define game $\mathsf{Game}^{\mathsf{(a)ke}-b}_{\alpha,\beta,\mathtt{P}}(\mathcal{A}, \kappa)$ as follows:*

- *after initialization $\mathcal{A}$ interacts with instance oracles using queries;*
- *if $\mathcal{A}$ asks a* Test *query to an α-fresh oracle Π_U^s which has accepted, it receives either $\mathtt{key}_1 := k_U^s$ (if $b = 1$) or $\mathtt{key}_0 \in_R \{0,1\}^\kappa$ (if $b = 0$);*

- *$\mathcal{A}$ continues interacting with instance oracles;*
- *when $\mathcal{A}$ terminates, it outputs a bit trying to guess which case it was dealing with.*

The output of $\mathcal{A}$ is the output of the game. The advantage function (over all adversaries running within time κ) in winning the game is defined as

$$\mathsf{Adv}^{\mathsf{(a)ke}}_{\alpha,\beta,\mathtt{P}}(\kappa) := \max_{\mathcal{A}} \left|2\Pr\left[\mathsf{Game}^{\mathsf{(a)ke}-b}_{\alpha,\beta,\mathtt{P}}(\mathcal{A},\kappa) = b\right] - 1\right|$$

We say that $\mathtt{P}$ is an (A)KE-secure protocol with α-secrecy*, denoted* $(\mathtt{A})\mathtt{GKE}$-α*, if the advantage* $\mathsf{Adv}^{\mathsf{(a)ke}}_{\alpha,\beta,\mathtt{P}}(\kappa)$ *is negligible. If $\alpha = \emptyset$, we just say that $\mathtt{P}$ is* (A)KE-secure.

Definition 6 (MA-Security). *Let $\mathtt{P}$ be a correct GKE protocol and* $\mathsf{Game}^{\mathsf{ma}}_{\mathtt{P}}(\mathcal{A},\kappa)$ *the interaction between the instance oracles and an active adversary $\mathcal{A}$ which can query* Send*,* Setup*,* RevealKey*,* RevealState*, and* Corrupt*. We say that $\mathcal{A}$* wins *if at some point during the interaction there exist an uncorrupted user U_i whose instance oracle $\Pi_i^{s_i}$ has accepted with $k_i^{s_i}$ and another user U_j with $U_j \in \mathtt{pid}_i^{s_i}$ that is uncorrupted at the time $\Pi_i^{s_i}$ accepts, such that*

1. *there exists **no** instance oracle $\Pi_j^{s_j}$ with $(\mathtt{pid}_j^{s_j}, \mathtt{sid}_j^{s_j}) = (\mathtt{pid}_i^{s_i}, \mathtt{sid}_i^{s_i})$, **or***
2. *there exists **an** instance oracle $\Pi_j^{s_j}$ with $(\mathtt{pid}_j^{s_j}, \mathtt{sid}_j^{s_j}) = (\mathtt{pid}_i^{s_i}, \mathtt{sid}_i^{s_i})$ that accepted with $k_j^{s_j} \neq k_i^{s_i}$.*

The maximum probability of this event (over all adversaries running within time κ) is denoted $\mathsf{Succ}^{\mathsf{ma}}_{\mathtt{P}}(\kappa)$*. We say that a GKE protocol $\mathtt{P}$ is* MA-secure *(*$\mathtt{MAGKE}$*) if this probability is a negligible function of κ.*

Note that U_i and U_j must be uncorrupted, however, $\mathcal{A}$ is allowed to reveal internal states of their oracles. Hence, our MA-security definition seems to be stronger than definitions of security against insider attacks in the KS model.

4 Compiler for AKE- and MA-Security in the Standard Model

4.1 Security-Enhancing Compilers and Their Goals

Imagine, there exists a *black-box* implementation of a GKE protocol which should be used by some group application and assume this implementation does not satisfy all security requirements desired for the group application. Instead of designing and implementing a new GKE protocol in an *ad-hoc* fashion, it is desirable to have a generic technique which can be applied to the given *black-box* implementation in order to enhance its security.

Obviously, a good strategy (though not always optimal) is to implement a GKE protocol in a modular way: one starts with the basic implementation that satisfies the most common set of security requirements, then continues with the implementation of optional modules that can be added to provide extended security requirements. The main goal of security-enhancing GKE protocol compilers is to enable secure construction of GKE protocols in such a modular way.

Definition 7 (Security-Enhancing GKE Protocol Compiler $\mathtt{C}$). *A* security-enhancing GKE compiler $\mathtt{C}$ *is a procedure which takes as input a GKE protocol $\mathtt{P}$ and outputs a compiled GKE protocol $\mathtt{C}_{\mathtt{P}}$ with additional security properties possibly missing in $\mathtt{P}$.*

4.2 Discussion on Existing Compilers for GKE Protocols

In the following we provide some analysis on currently known security-enhancing compilers for GKE protocols.

Compiler for AKE-Security. KE-security, i.e., key indistinguishability with respect to passive adversaries is the basic security requirement for a GKE protocol. Authentication may be optional if a network or a high-level application already provides it. Seeing AKE as an additional property, Katz and Yung designed the corresponding compiler in [31].

Definition 8 (Compiler for AKE-Security [31])). *Let* $\mathtt{P}$ *be a GKE protocol and* $\Sigma := (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ *a digital signature scheme. A* compiler for AKE-security *consists of an initialization algorithm and a modified protocol execution:*

`Initialization`: *each* $U_i \in \mathcal{U}$ *generates his own additional private/public key pair* (sk'_i, pk'_i) *using* $\Sigma.\mathtt{Gen}(1^{\kappa'})$.

`The protocol`: *prior to any operation execution of* $\mathtt{P}$*:*

- *Each* Π_i^s *chooses a nonce* $r_i \in_R \{0,1\}^\kappa$ *and sends* $U_i|0|r_i$ *to its partners.*
- *After* Π_i^s *receives the* $U_j|0|r_j$*'s it computes* $\mathtt{sid}_i^s := U_1|r_1| \ldots |U_n|r_n$.

Then, members of $\mathcal{G}$ *execute* $\mathtt{P}$ *with the following changes:*

- *When* Π_i^s *is supposed to send a message* $U_i|t|m$*, it computes* $\sigma_i := \Sigma.\mathtt{Sign}(sk'_i, t|m|\mathtt{sid}_i^s)$ *and sends the modified message* $U_i|t|m|\sigma_i$.
- *When* Π_i^s *receives* $U_j|t|m|\sigma_j$ *it checks whether (1)* $U_j \in \mathtt{pid}_i^s$*, (2)* t *is the next expected sequence number, and (3)* $\Sigma.\mathtt{Verify}(pk'_j,\ t|m|\mathtt{sid}_i^s,\ \sigma_j) =?1$. *If all checks pass, it proceeds as in* $\mathtt{P}$ *upon receiving* $U_j|t|m$.
- *After* Π_i^s *computes the session key* k_i^s *in* $\mathtt{P}$*, it accepts with this key.*

Missing Generality of Katz-Yung Compiler. Katz and Yung proved security of this compiler assuming an unreliable asynchronous broadcast channel and a passive adversary, which is only an *eavesdropper*. We show that in this case their compiler is not really generic: there exist GKE protocols that are secure against eavesdroppers but become insecure against active adversaries (even after the execution of the above compiler).

We consider the following (pathologic, but illustrative) protocol between Π_1^s and Π_1^s. First Π_1^s chooses his exponent $x_1 \in_R \mathbb{Z}_q^\star$ and sends $X_1 := g^{x_1}$ to Π_2^s. If Π_2^s receives X_1 within some specified time period δ then Π_2^s replies with $X_2 := g^{x_2}$ for some randomly chosen $x_2 \in_R \mathbb{Z}_q^\star$ and accepts with the Diffie-Hellman session key $g^{x_1 x_2}$. Similar if Π_1^s receives X_2 within time δ then it accepts with $g^{x_1 x_2}$ too. However, if an oracle does not receive data in time, it accepts with g. If the passive adversary is just an eavesdropper, messages are delivered on time, and the protocol is "passively" secure. But an active adversary can drop messages so that both participants accept with g. Obviously, it is insufficient to restrict passive adversaries to be just eavesdroppers. Passive attacks should also model the unreliability of the communication, like we do.

Compilers for MA-Security. The first compiler for key confirmation and mutual authentication was proposed by Bresson *et al.* [14]. However, their definition of MA is (*de facto*) the old one, and the proof is conducted in the Random Oracle Model [6].

Katz and Shin [30] showed how to turn an AKE-secure GKE protocol into a UC-secure GKE protocol that provides security against insider attacks (see Section 2.2).

Definition 9 (Compiler for Security against Insider Attacks by Katz and Shin). *Let* $\mathtt{P}$ *be a GKE protocol,* $\Sigma := (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ *a digital signature scheme,* $F := \{f_k\}_{k\in\{0,1\}^\kappa}$ *a function ensemble with range* $\{0,1\}^\lambda$, $\lambda \in \mathbb{N}$ *and domain* $\{0,1\}^\kappa$*, and* $\mathtt{sid}_i^s$ *is a unique session id. A* compiler for security against insider attacks *consists of an initialization algorithm and a protocol defined as follows:*

$\mathtt{Initialization}$: *each* $U_i \in \mathcal{U}$ *generates his own additional private/public key pair* (sk'_i, pk'_i) *using* $\Sigma.\mathtt{Gen}(1^{\kappa'})$.

$\mathtt{The\ protocol}$: *After an oracle* Π_i^s *accepts with* $(k_i^s, \mathtt{pid}_i^s, \mathtt{sid}_i^s)$ *in* $\mathtt{P}$*:*

- *it computes* $\mu_i := f_{k_i^s}(v_0)$ *where* v_0 *is a constant public value;*
- *it computes* $K_i^s := f_{k_i^s}(v_1)$ *where* $v_1 \neq v_0$ *is another constant public value;*
- *it erases its local state information except for* μ_i, K_i^s, $\mathtt{pid}_i^s$*, and* $\mathtt{sid}_i^s$*;*
- *it computes a signature* $\sigma_i := \Sigma.\mathtt{Sign}(sk'_i,\ \mu_i|\mathtt{sid}_i^s|\mathtt{pid}_i^s)$ *and sends* $U_i|\sigma_i$ *to its partnered oracle* Π_j^s.

After $\Pi_i^{s_i}$ *receives* $U_j|\sigma_j$ *from all its partnered oracle* Π_j^s*:*

- *it checks whether* $\Sigma.\mathtt{Verify}(pk'_j, \mu_i|\mathtt{sid}_i^s|\mathtt{pid}_i^s, \sigma_j) \stackrel{?}{=} 1$*;*
- *if all checks pass, it accepts with the session key* K_i^s.

4.3 Compiler C-AMA

In the following we describe a compiler (denoted C-AMA) which provides both AKE- and MA-security. C-AMA uses nonces to achieve uniqueness of protocol sessions and security of concurrent executions without relying on session ids given by high-level applications.

Definition 10 (Compiler for AKE- and MA-Security C-AMA). *Let* $\mathtt{P}$ *be a GKE protocol,* $\Sigma := (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ *a digital signature scheme,* $F := \{f_k\}_{k\in\{0,1\}^\kappa}$ *a function ensemble with range* $\{0,1\}^\lambda$, $\lambda \in \mathbb{N}$ *and domain* $\{0,1\}^\kappa$*. A* compiler for AKE-security and MA-security*, denoted* C-AMA*, consists of an algorithm* INIT *and a protocol* AMA *defined as follows:*

INIT: *each* $U_i \in \mathcal{U}$ *generates own private/public key pair* (sk'_i, pk'_i) *using* $\Sigma.\mathtt{Gen}(1^{\kappa'})$.

AMA: *prior to the execution of* $\mathtt{P}$*:*

- *Each* Π_i^s *chooses a* AMA nonce $r_i \in_R \{0,1\}^\kappa$ *and sends* $U_i|r_i$ *to its partners.*
- *After* Π_i^s *receives all* $U_j|r_j$*, it computes* $\mathtt{sid}_i^s := r_1|\ldots|r_n$.

Then it invokes the execution of $\mathtt{P}$ *and proceeds as follows:*

- *If* Π_i^s *in* $\mathtt{P}$ *outputs a message* $U_i|m$ *then in* $\mathtt{C\text{-}AMA}_{\mathtt{P}}$ *it computes additionally* $\sigma_i := \Sigma.\mathtt{Sign}(sk'_i,\ m|\mathtt{sid}_i^s|\mathtt{pid}_i^s)$ *and outputs a modified message* $U_i|m|\sigma_i$.
- *On receiving* $U_j|m|\sigma_j$ *from a partner* Π_j^s *it checks whether* $\Sigma.\mathtt{Verify}(pk'_j,\ m|\mathtt{sid}_i^s|\mathtt{pid}_i^s,\ \sigma_j) \stackrel{?}{=} 1$*. If this fails then* Π_i^s *terminates; otherwise it proceeds as in* $\mathtt{P}$ *upon receiving* $U_j|m$.
- *After an oracle* Π_i^s *computes* k_i^s *in* $\mathtt{P}$ *it computes an* AMA token $\mu_i := f_{k_i^s}(v_0)$ *where* v_0 *is a constant public value, a signature* $\sigma_i := \Sigma.\mathtt{Sign}(sk'_i,\ \mu_i|\mathtt{sid}_i^s|\mathtt{pid}_i^s)$ *and sends* $U_i|\sigma_i$ *to every* Π_j^s *with* $U_j \in \mathtt{pid}_i^s$.
- *On receiving* $U_j|\sigma_j$ *from its partner,* Π_i^s *checks if* $\Sigma.\mathtt{Verify}(pk'_j, \mu_i|\mathtt{sid}_i^s|\mathtt{pid}_i^s, \sigma_j) \stackrel{?}{=} 1$.
- *If all checks pass,* Π_i^s *computes* $K_i^s := f_{k_i^s}(v_1)$ *where* $v_1 \neq v_0$ *is another constant public value, erases all private state information from* $\mathtt{state}_i^s$ *(including* k_i^s*) and accepts with* K_i^s.

PERFORMANCE ANALYSIS. C-AMA requires two further rounds: one to exchange random nonces and another one to exchange signatures on AMA tokens. As for the computation costs, every participant generates one signature for every message sent during P and one additional signature on the computed AMA token. Furthermore, every participant must verify one signature for every incoming message in P and $n-1$ signatures during the additional confirmation round. The computation of the AMA token μ and of the session key K can be seen as negligible.

SECURITY ANALYSIS. Our first theorem (full proof will be given in the full version) shows that C-AMA adds AKE-security to any KE-secure GKE protocol. Following Remark 4, we do not consider the adversarial setting $(\mathtt{sbs}, \mathtt{scm})$. For the definition of collision-resistance for F we refer for example to [30].

Theorem 11 (AKE-Security of C-AMA$_\mathtt{P}$). *Let* $(\alpha, \beta) \in \{(\emptyset, \mathtt{wcm}), (\mathtt{wbs}, \mathtt{wcm\text{-}bs}), (\mathtt{wfs}, \mathtt{wcm\text{-}fs}), (\mathtt{sfs}, \mathtt{scm})\}$ *be an adversarial setting, let* P *be a* GKE-α *protocol, and* $\mathcal{A}$ *an active adversary in the corruption model* β *launching at most* q_s *sessions of* C-AMA$_\mathtt{P}$. *If* Σ *is EUF-CMA and* F *is pseudo-random then* C-AMA$_\mathtt{P}$ *is* AGKE-α, *and*

$$\mathsf{Adv}^{\mathsf{ake}}_{\alpha,\beta,\mathtt{C\text{-}AMA}_\mathtt{P}}(\kappa) \leq 2N\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa) + \frac{Nq_\mathsf{s}^2}{2^{\kappa-1}} + 2q_\mathsf{s}\mathsf{Adv}^{\mathsf{ke}}_{\alpha,\beta,\mathtt{P}}(\kappa) + 4q_\mathsf{s}\mathsf{Adv}^{\mathsf{prf}}_{F}(\kappa).$$

Proof (Sketch). We define a *sequence of games* [40] $\mathbf{G}_i$, $i = 0, \ldots, 6$ and denote by $\mathsf{Win}^{\mathsf{ake}}_i$ the event that the adversary correctly guesses the bit b in $\mathbf{G}_i$.

Game $\mathbf{G}_0$: This is the real $\mathsf{Game}^{\mathsf{ake}-b}_{\alpha,\beta,\mathtt{C\text{-}AMA}_\mathtt{P}}(\mathcal{A}, \kappa)$ played between the simulator $\mathcal{S}$ and an active adversary $\mathcal{A}$.

Game $\mathbf{G}_1$: Here the simulation fails and bit b' is set at random if $\mathcal{A}$ asks a Send query on some $U_i|m|\sigma$ (or $U_i|\sigma$) such that σ is a valid signature that has not been previously output by an oracle Π_i^s before querying $\mathsf{Corrupt}(U_i)$, i.e., if a forgery occurs. One can show that $|\Pr[\mathsf{Win}^{\mathsf{ake}}_1] - \Pr[\mathsf{Win}^{\mathsf{ake}}_0]| \leq N\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa)$.

Game $\mathbf{G}_2$: In this game the simulation fails and bit b' is set at random if an AMA nonce r_i is used by any uncorrupted user's oracle Π_i^s in two different sessions. Considering collisions for the choice of AMA nonces one can show that $|\Pr[\mathsf{Win}^{\mathsf{ake}}_2] - \Pr[\mathsf{Win}^{\mathsf{ake}}_1]| \leq \frac{Nq_\mathsf{s}^2}{2^\kappa}$. This game excludes replay attacks.

Game $\mathbf{G}_3$: In this game we add the following rule: $\mathcal{S}$ chooses $q_\mathsf{s}^* \in [1, q_\mathsf{s}]$ as a guess for the number of sessions invoked before $\mathcal{A}$ asks the query Test. If this query does not occur in the q_s^*-th session then the simulation fails and bit b' is set at random. One can show that $\Pr[\mathsf{Win}^{\mathsf{ake}}_2] = q_\mathsf{s}\left(\Pr[\mathsf{Win}^{\mathsf{ake}}_3] - \frac{1}{2}\right) + \frac{1}{2}$.

Game $\mathbf{G}_4$: In this game $\mathcal{S}$ acts as a passive adversary against the KE-security of P that participates in $\mathsf{Game}^{\mathsf{ke}-1}_{\alpha,\beta,\mathtt{P}}(\kappa)$, i.e., the Test query of $\mathcal{S}$ to an accepted α-fresh oracle Π_i^s in P is answered with the real session group key k_i^s. In the full version we show how $\mathcal{S}$ answers the queries of $\mathcal{A}$ such that $\Pr[\mathsf{Win}^{\mathsf{ake}}_4] = \Pr[\mathsf{Win}^{\mathsf{ake}}_3]$.

Game $\mathbf{G}_5$: The only difference to $\mathbf{G}_4$ is that $\mathcal{S}$ participates as a passive adversary in $\mathsf{Game}^{\mathsf{ke}-0}_{\alpha,\beta,\mathtt{P}}(\kappa)$, i.e., the Test query is answered with a random bit string. Hence, $|\Pr[\mathsf{Win}^{\mathsf{ake}}_5] - \Pr[\mathsf{Win}^{\mathsf{ake}}_4]| \leq \mathsf{Adv}^{\mathsf{ke}}_{\alpha,\beta,\mathtt{P}}(\kappa)$.

Game $\mathbf{G}_6$: This game is identical to $\mathbf{G}_5$ except that in the q_s^*-th session K and the AMA token μ are replaced by random values sampled from $\{0,1\}^\kappa$. Hence, $|\Pr[\mathsf{Win}_6^{\mathsf{ake}}] - \Pr[\mathsf{Win}_5^{\mathsf{ake}}]| \leq 2\mathsf{Adv}_F^{\mathsf{prf}}(\kappa)$ and $\Pr[\mathsf{Win}_6^{\mathsf{ake}}] = \frac{1}{2}$. This results in the desired inequality. □

Our next theorem shows that C-AMA provides MA-security (with malicious participants and strong corruptions) for any GKE protocol P.

Theorem 12 (MA-Security of C-AMA$_\mathtt{P}$). *Let* P *be a GKE protocol and $\mathcal{A}$ an active adversary launching at most q_s sessions of* C-AMA$_\mathtt{P}$*. If Σ is EUF-CMA and F is collision-resistant then* C-AMA$_\mathtt{P}$ *is* MAGKE*, and*

$$\mathsf{Succ}_{\texttt{C-AMA}_\mathtt{P}}^{\mathsf{ma}}(\kappa) \leq N\mathsf{Succ}_\Sigma^{\mathtt{euf-cma}}(\kappa) + \frac{Nq_s^2}{2^\kappa} + q_s\mathsf{Succ}_F^{\mathsf{coll}}(\kappa).$$

Proof (Sketch). We define a *sequence of games* $\mathbf{G}_i$, $i = 0, \ldots, 2$ and events $\mathsf{Win}_i^{\mathsf{ma}}$ meaning that $\mathcal{A}$ wins in $\mathbf{G}_i$. $\mathcal{A}$'s queries are answered by a simulator $\mathcal{S}$.

Game $\mathbf{G}_0$: This is the real $\mathsf{Game}_{\texttt{C-AMA}_\mathtt{P}}^{\mathsf{ma}}(\mathcal{A}, \kappa)$ played between $\mathcal{S}$ and $\mathcal{A}$. The goal of $\mathcal{A}$ is to achieve that there exists an uncorrupted user U_i whose corresponding oracle Π_i^s accepts with K_i^s and another user $U_j \in \mathtt{pid}_i^s$ that is uncorrupted at the time Π_i^s accepts and either does not have a corresponding oracle Π_j^s with $(\mathtt{pid}_j^s, \mathtt{sid}_j^s) = (\mathtt{pid}_i^s, \mathtt{sid}_i^s)$ or has such an oracle but this oracle accepts with $K_j^s \neq K_i^s$.

Game $\mathbf{G}_1$: Here the simulation fails if a forgery occurs. Similar to the previous proof we have: $|\Pr[\mathsf{Win}_1^{\mathsf{ma}}] - \Pr[\mathsf{Win}_0^{\mathsf{ma}}]| \leq N\mathsf{Succ}_\Sigma^{\mathtt{euf-cma}}(\kappa)$.

Game $\mathbf{G}_2$: Her we abort in case of nonces collision. $|\Pr[\mathsf{Win}_2^{\mathsf{ma}}] - \Pr[\mathsf{Win}_1^{\mathsf{ma}}]| \leq \frac{Nq_s^2}{2^\kappa}$.

Having excluded forgeries and replay attacks, we follow that for every $U_j \in \mathtt{pid}_i^s$ that is uncorrupted when Π_i^s accepts, there exists a corresponding Π_j^s with $(\mathtt{pid}_j^s, \mathtt{sid}_j^s) = (\mathtt{pid}_i^s, \mathtt{sid}_i^s)$. Thus, according to Definition 6 $\mathcal{A}$ wins in this game only if any of these oracles has accepted with $K_j^s \neq K_i^s$. Arguing by contradiction one can show that $\Pr[\mathsf{Win}_2^{\mathsf{ma}}] = \Pr[K_i^s \neq K_j^s \wedge f_{k_i^s}(v_0) = f_{k_j^s}(v_0)] \leq q_s\mathsf{Succ}_F^{\mathsf{coll}}(\kappa)$ and obtain the desired inequality. □

5 Conclusion

In this paper we found some problems with the definition of MA-security in the foundational BCPQ model. We proposed a revised definition which considers malicious participants and unifies many of the well-known informal notions. Additionally, we extended the (strong) forward secrecy in AKE-security by the symmetrically opposed notion of (strong) backward secrecy. Further we described the provably secure generic compiler C-AMA that adds AKE- and MA-security to any GKE protocol which is passively secure (wrt. to an adversary which can change the delivery order of messages, and delay or drop them).

We refer to our parallel work in [15,35] for further development, in particular, in the light of the *contributory* nature of GKE protocols and the appropriate generic solution for this security goal.

References

1. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-Based Group Key Exchange in a Constant Number of Rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
2. Ateniese, G., Steiner, M., Tsudik, G.: Authenticated Group Key Agreement and Friends. In: CCS 1998, pp. 17–26 (1998)
3. Barak, B., Lindell, Y., Rabin, T.: Protocol Initialization for the Framework of Universal Composability. http://eprint.iacr.org/2004/006
4. Bellare, M.: Practice-Oriented Provable-Security. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 221–231. Springer, Heidelberg (1998)
5. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
6. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: CCS 1993, pp. 62–73 (1993)
7. Bellare, M., Rogaway, P.: Provably Secure Session Key Distribution: The Three Party Case. In: STOC 1995, pp. 57–66 (1995)
8. Bohli, J.-M., Vasco, M.I.G., Steinwandt, R.: Secure Group Key Establishment Revisited. http://eprint.iacr.org/2005/395
9. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer, Heidelberg (2003)
10. Boyd, C., Nieto, J.M.: Round-Optimal Contributory Conference Key Agreement. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 161–174. Springer, Heidelberg (2002)
11. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290–390. Springer, Heidelberg (2001)
12. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
13. Bresson, E., Chevassut, O., Pointcheval, D.: Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 497–514. Springer, Heidelberg (2002)
14. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably Authenticated Group Diffie-Hellman Key Exchange. In: CCS 2001, pp. 255–264 (2001)
15. Bresson, E., Manulis, M.: Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 395–409. Springer, Heidelberg (2007)
16. Burmester, M.: On the Risk of Opening Distributed Keys. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 308–317. Springer, Heidelberg (1994)
17. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
18. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS 2001, pp. 136–145 (2001)
19. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
20. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Errors in Computational Complexity Proofs for Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)

21. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
22. Desmedt, Y., Pieprzyk, J., Steinfeld, R., Wang, H.: A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders. In: Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 459–475. Springer, Heidelberg (2006)
23. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Trans. on Information Theory IT-22(6), 644–654 (1976)
24. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and Authenticated Key Exchanges. Designs, Codes and Cryptography 2(2), 107–125 (1992)
25. Dutta, R., Barua, R.: Constant Round Dynamic Group Key Agreement. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 74–88. Springer, Heidelberg (2005)
26. Dutta, R., Barua, R.: Dynamic Group Key Agreement in Tree-Based Setting. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 101–112. Springer, Heidelberg (2005)
27. Dutta, R., Barua, R., Sarkar, P.: Provably Secure Authenticated Tree Based Group Key Agreement. In: Lopez, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 92–104. Springer, Heidelberg (2004)
28. Günther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
29. Ingemarsson, I., Tang, D.T., Wong, C.K.: A Conference Key Distribution System. IEEE Trans. on Information Theory 28(5), 714–719 (1982)
30. Katz, J., Shin, J.S.: Modeling Insider Attacks on Group Key-Exchange Protocols. In: CCS 2005, pp. 180–189 (2005)
31. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
32. Kim, H.-J., Lee, S.-M., Lee, D.H.: Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 245–259. Springer, Heidelberg (2004)
33. Kim, Y., Perrig, A., Tsudik, G.: Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups. In: CCS 2000, pp. 235–244 (2000)
34. Kim, Y., Perrig, A., Tsudik, G.: Communication-Efficient Group Key Agreement. In: IFIP/Sec 2001, pp. 229–244 (2001)
35. Manulis, M.: Provably Secure Group Key Exchange. PhD thesis, Ruhr University Bochum,(June 2007)
36. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press,(October 1996)
37. Pereira, O., Quisquater, J.-J.: Some Attacks upon Authenticated Group Key Agreement Protocols. J. of Computer Security 11(4), 555–580 (2003)
38. Perrig, A.: Efficient Collaborative Key Management Protocols for Secure Autonomous Group Communication. In: CryptEC 1999, pp. 192–202 (1999)
39. Shoup, V.: On Formal Models for Secure Key Exchange (Version 4). IBM RZ 3120 (November 1999), http://shoup.net/
40. Shoup, V.: Sequences of Games: A Tool for Taming Complexity in Security Proofs. http://eprint.iacr.org/2004/332
41. Steer, D.G., Strawczynski, L., Diffie, W., Wiener, M.J.: A Secure Audio Teleconf. System. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 520–528. Springer, Heidelberg (1990)
42. Steiner, M., Tsudik, G., Waidner, M.: CLIQUES: A New Approach to Group Key Agreement. In: ICDCS 1998, pp. 380–387 (1998)
43. Yacobi, Y., Shmuely, Z.: On Key Distribution Systems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 344–355. Springer, Heidelberg (1990)

Processing Multi-parameter Attacktrees with Estimated Parameter Values

Aivo Jürgenson[1,2] and Jan Willemson[3,4]

[1] Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia
aivo.jurgenson@eesti.ee
[2] Elion Enterprises Ltd, Endla 16, 15033 Tallinn, Estonia
[3] Tartu University, Institute of Computer Science, Liivi 2, Tartu, Estonia
jan@ut.ee
[4] Cybernetica, Aleksandri 8a, Tartu, Estonia

Abstract. Authors extend the multi-parameter attacktree model to include inaccurate or estimated parameter values, which are modelled as probabilistic interval estimations. The paper develops mathematical tools to extend the computation rules of the attacktree model to work with interval estimations instead of point estimates. We present a sample computation routine and discuss how to interpret the analysis results and how to choose the optimal or an economically justified security level.

1 Introduction

Recent developments in information technology have changed the way we live and work. We can communicate faster and in larger volumes than ever before, our productivity has increased dramatically due to task automation and parallelization, etc. Unfortunately, information technology has also helped the dark side. Besides legitimate work, attacking someone's (digital) assets has become much more efficient as well. Properties of the digital world make attacks highly parallelizable, the traces easily hidable and the damage occurring almost instantly.

Thus, in order to perform one's duties under such conditions, applying adequate security mechanisms becomes a necessary prerequisite. Still, the number of possible attack countermeasures is large and the task of picking the right set is far from being trivial. Accordingly, there are several approaches for this task.

Parker [1] emphasizes the importance of planning and selecting information safeguards to achieve due diligence toward achieving enablement of trustworthy business and engineering systems and compliance with the regulations and legislation. For Parker, compliance is nowadays more important than security risk reduction considering the penalties being applied to organizations that fail to meet regulatory requirements.

The authors of the current paper however feel that even though compliance to regulatory requirements may ensure a *sufficient* security level, several aspects of security management remain uncovered. For example, business management usually does not only require security level to be sufficient, but also *optimal* in the sense that no over exaggerated investments have been made. Thus, a good security manager must be able

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 308–319, 2007.
© Springer-Verlag Berlin Heidelberg 2007

to explain to the board, what the company is getting in return for the money invested into security [2,3].

This question is very hard to answer without a thorough risk assessment. Thus, developing methods for IT risk analysis is a major challenge requiring a solution when building large computer-dependent infrastructures.

Since attacks are human created and constantly evolving, it is not possible to establish any fully automatic risk analysis mechanisms (even though several promising approaches exist based on attack graphs [4,5]). Expert knowledge will always play a substantial role with security analysis. However, expert evaluations are generally rather rough and can not cope with very complicated threats. For instance, [6] provides us with approximate expert-assigned probabilities of a number of threats with the precision of 0.1 on the scale $0 \ldots 1$. Even with such level of roughness, we only see estimates for relatively simple events "Attempted Unauthorized System Access by Outsider", "Abuse of Access Privileges by Other Authorized User", etc., but not for complicated scenarios like "Loss of profits due to lost user base after online service inaccessibility for 5 hours".

Thus a method needs to be developed that on one hand is able to handle complicated threats, but at the same time could still make use of approximate expert knowledge. In this paper we study a method related to attack graphs as considered by Sheyner et. al. [5,7,8], where one is interested in describing and comparing different event sequences that will eventually result in successful penetration of security mechanisms. However, we will assume some extra structure from these graphs so that different subattacks are organized hierarchically and form a tree. The next Section will cover this issue in more detail.

2 Hierarchical Security Assessment – State of the Art

In order to better assess security level of a complex and heterogeneous system, a gradual refinement method called *threat tree* or *attack tree method* can be used. Basic idea of the approach is simple – the analysis begins by identifying one or more *primary threats* and continues by splitting the threat into subattacks, either all or some of them being necessary to materialize the primary threat. The subattacks can be split further etc., until we reach the state where it no more makes sense to split the resulting attacks any further; these kinds of non-splittable attacks are called *elementary* or *atomic attacks* and the security analyst will have to evaluate them somehow. During the splitting process, a tree is formed having the primary threat in its root and elementary attacks in its leaves. Using the structure of the tree and the estimations of the leaves, it is then (hopefully) possible to give some estimations of the root node as well.

Threat tree approach to security evaluation is several decades old. It has been used for tasks like fault assessment of critical systems [9] or software vulnerability analysis [10,11], and was adapted to information security by Bruce Schneier [12,13].

Earlier works in this field considered attack trees using only one estimated parameter like cost or feasibility of the attack, skill level required, etc. [11,12,14]. Opel [15] considered also multi-parameter attack trees, but the actual tree computations in his model still used only one parameter at a time. Even though single-parameter attack trees can capture some aspects of threats reasonably well, they still lack the ability to describe the full complexity of the attacker's decision-making process.

A substantial step towards better understanding the motivation of the attacker was made in 2006 by Buldas et. al. [16]. Besides considering just the cost of the attack, they also used success probability together with probabilities and amount of penalties in the case of success or failure of the attack in their analysis. As a result, a much more accurate model of the attack game was obtained.

The model of [16] has a significant drawback when it comes to practical application. Namely, the authors of [16] consider all the parameter values to be precise point estimates. Still, in practice security analysts rarely tend to provide exact numerical values for costs, probabilities etc. Instead, it feels much more natural to talk about intervals where the parameters belong to with some confidence.

The purpose of this paper is to extend the research of Buldas et. al. [16] by replacing exact values with interval estimations. The paper is organized as follows. First, in Section 3 we give a more formal definition of attack trees. In order to be able to give estimations of higher level attacks based on more elementary ones, rules of computation with interval estimations must be developed and this is done in Section 4. Section 5 presents an example of the computation routine and gives general rules for result interpretation. Finally, in Section 6 we draw some conclusions and give directions for future work.

3 Attack Trees

As mentioned in Section 2, attack tree is a result of a top-down process of splitting complex attacks into simpler ones. In this paper, we will consider two types of splits – AND-splits and OR-splits.[1] Thus, there are altogether three types of nodes in the tree.

1. *Leaf node* or *elementary attack*, which does not have any subattacks and which success does not depend on any other attacks. The parameter values of the leaf nodes are assigned by the experts.
2. *OR-node*, which has child nodes; for the OR-node attack to succeed, at least one of the sub-attacks must succeed.
3. *AND-node*, which has child nodes; for the AND-node attack to succeed, all of the sub-attacks must succeed as well.

Following [16], we will use the parameters in Table 1 that are to be evaluated in the leaf nodes and computed throughout the tree.

It will later prove useful to denote the expected loss in case the attack was successful $q_S \cdot k_S$ by π_S and the expected loss in case the attack was not successful $q_F \cdot k_F$ by π_F.

We will denote the cost of the elementary attack A as $\mathsf{Cost}(A)$ and similar notation will be used for other parameters as well.

We will later use the example attack tree given in Figure 1 describing a simple security analysis of information leak from a company. The tree has four leaf nodes, two

[1] Even though the approach using only AND and OR splits is not flexible enough to cover all possible security settings (e.g. threshold security), they have proven to be enough in all practical threat trees analyzed by the authors.

Table 1. Parameters of the attacks

Parameter	Definition
Cost	cost of performing the attack
p	probability of success of the attack
q_S	probability of catching the attacker, if the attack was successful
k_S	penalty of the attacker, if the attack was successful and attacker was caught
q_F	probability of catching the attacker, if the attack was not successful
k_F	penalty of the attacker, if the attack was not successful and attacker was caught

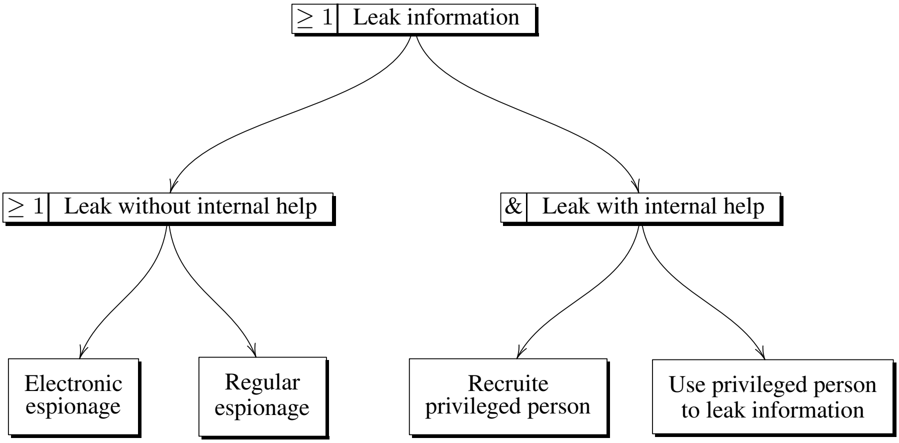

≥ 1 Leak information
≥ 1 Leak without internal help
& Leak with internal help
Electronic espionage
Regular espionage
Recruite privileged person
Use privileged person to leak information

Fig. 1. Attacktree of leaking sensitive information from a company

OR-nodes (one of them denoting the primary threat at the root) and one AND-node. We are using notion similar to AND-gates ($\boxed{\&}$) and OR-gates ($\boxed{\geq 1}$) to distinguish between AND-nodes and OR-nodes.

3.1 Tree Computations

The authors of [16] give the following formulae for computing the parameters of parent node C based on the values of child nodes A and B. If C is an AND-node, we get

$$\mathsf{Cost}(C) = \mathsf{Cost}(A) + \mathsf{Cost}(B), \tag{1}$$

$$p(C) = p(A) \cdot p(B), \tag{2}$$

$$\pi_S(C) = \pi_S(A) + \pi_S(B), \tag{3}$$

$$\begin{aligned}\pi_F(C) = \tfrac{1}{1-p(A)p(B)} \cdot [&p(A)(1-p(B))(\pi_S(A) + \pi_F(B)) + \\ &+(1-p(A))p(B)(\pi_F(A) + \pi_S(B)) + \\ &+(1-p(A))(1-p(B))(\pi_F(A) + \pi_F(B))].\end{aligned} \tag{4}$$

The following formula is used in case the node C is an OR-node.

$$
\begin{aligned}
&(\mathsf{Cost}(C), p(C), \pi_S(C), \pi_F(C)) \\
&= \begin{cases} (\mathsf{Cost}(A), p(A), \pi_S(A), \pi_F(A)) & \text{if } \mathsf{Outcome}(A) \geq \mathsf{Outcome}(B) \\ (\mathsf{Cost}(B), p(B), \pi_S(B), \pi_F(B)) & \text{if } \mathsf{Outcome}(A) < \mathsf{Outcome}(B) \end{cases} ,
\end{aligned} \tag{5}
$$

where $\mathsf{Outcome}(A)$ is the outcome of the attack A for attacker. Its value is computed as

$$
\mathsf{Outcome}(A) = p(A) \cdot \mathsf{Gain} - p(A) \cdot \pi_S(A) - (1 - p(A)) \cdot \pi_F(A) - \mathsf{Cost}(A) , \tag{6}
$$

where Gain is a global parameter expressing the total gain of the attacker in case the primary threat is materialized.

4 Modeling Parameter Estimations

As discussed above, security experts may find it more comfortable working with intervals, rather than exact values. When talking about the value of some parameter belonging to an interval, such claims are usually not absolute, but hold with some level of confidence. So we can write

$$
p_X = \Pr[k_1 \leq X \leq k_2] , \tag{7}
$$

where p_X is the *probability* of the unknown value of the parameter X being within the interval of $[k_1, k_2]$. We will later refer to p_X as *confidence* or *confidence level* and $\mathcal{X} = (p_X, k_1, k_2)$ as *estimation*. The set of all estimations will be denoted as $\mathbb{P}$.

4.1 Motivation and Connections with Bayesian Networks

In order to handle the estimations in intuitively comprehensible manner, we will consider estimated parameters as random variables. The probabilistic inference between random variables has been extensively studied in the Bayesian Networks (BN) theory and used e.g. in the fields of artificial intelligence and machine learning. Attack trees can be viewed as a special kind of BN graphs, which try to investigate the likelihood of the primary threats, given the information about leaf attacks. Attack tree structure represents causality connections between attacks and the node parameters represent random variables in the BN graph. One way to ”convert” attack trees to the causal networks has been presented by Qin and Lee [17]. In case of multi-parameter attack trees, each node has many variables and the inference between nodes is a bit more complex, as expressed by the formulae (1) – (6).

The general structure of operations on estimations follows a simple pattern – given argument estimations, we first convert them to random variables, then perform our computation operations and then convert the resulting random variable back to the estimation based on its distribution. In order to do the first conversion, we need some assumptions about the corresponding distributions, and in the current paper we will take all our distributions to be normal. Additionally, we assume that all our estimations in leaf nodes and therefore the random variables as well are independent of each other, which in practice is roughly usually the case. This allows us to simplify the operations on random variables.

The assumption about normal distributions is natural for most of the parameters, since security experts evaluating them are humans and humans tend to estimate values using normal distribution. The concept of using imprecise data and estimations in the BN graphs is not new and has been explored in [18] and [19]. However, in this paper, we do not try to compute the exact (conditional) distributions of all our variables, but simply use normal distributions as is generally done when trying to simplify the BN calculations. Note that we merely use the normality assumption as a heuristic that helps our expert to deduce estimations of parent nodes based on estimations of the child nodes in the attack tree, and we make no formal claims concerning what these distributions in reality might be.

One can think of the analyst's task to find out whether the system is vulnerable with respect to the primary threat. While completing this task, the analyst considers the leaves of his current tree (which just consists of one root node in the beginning of the process) and if (s)he is unable to evaluate some parameters of some leaf, (s)he must develop it further. After "solving" the resulting subtree, only the estimations of the parameters of the current node are important, and several heuristics can be used to achieve a reasonable result. The heuristic assuming normal distributions is just one possibility; other possibilities definitely exist, but they remain outside of the scope of the current paper.

There is another detail to note. Some of our parameters have fixed value domains, e.g. are probabilities and hence belong to $[0,1]$, so we can not claim that they follow normal distribution. However, considering probabilities itself as imprecise values (second-order probability distribution) is well known in Bayesian statistics and for our estimations it is enough that $[k_1,k_2] \subset [0,1]$. There is no harm caused when the corresponding parameter is internally interpreted as a normally distributed random variable, even if its original value represents a probability. We will cover this issue in more detail in the end of Section 4.

4.2 Estimation Arithmetic

Our goal is to replace exact parameter values in formulae (1) – (6) by estimations. To do so, we will have to define addition, subtraction, multiplication, division and comparison of estimations, but also multiplication by and adding to a real number.

To use estimations in our formulae, we next discuss how to define the required operations in such a way that $\mathbb{P}$ would become closed under these operations.

Conversion between estimations and random variables. To convert the estimation $\mathcal{X}$ to a random variable X, we have to find out the mean a_X and standard deviation σ_X. From the assumption above and from equation (7) we can get the following formulae:

$$a_X = \mathbf{E}X = \frac{k_1+k_2}{2} \quad , \tag{8}$$

$$p_X = \Pr(k_1 \le X \le k_2) = \Phi\left(\frac{k_2 - a_X}{\sigma_X}\right) - \Phi\left(\frac{k_1 - a_X}{\sigma_X}\right) \quad , \tag{9}$$

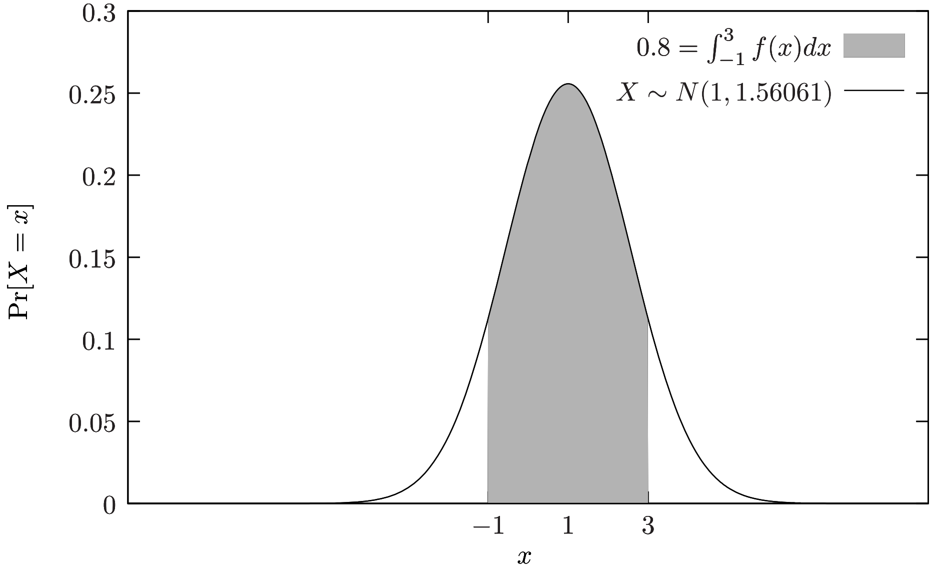


Fig. 2. Conversion $\mathcal{X} = (0.8, -1, 3) \to X \sim N(1, 1.56061)$

where the $\Phi(x)$ is the Laplace's function. Although we cannot give explicit formula for calculating σ_X, we can certainly solve the equation (9) to compute the standard deviation σ_X using a computer.

We denote conversion of estimation $\mathcal{X}$ to normally distributed random variable X as $\mathcal{X} = (p_X, k_1, k_2) \to X \sim N(a_X, \sigma_X)$. An example is depicted in Figure 2 for the conversion $\mathcal{X} = (0.8, -1, 3) \to X \sim N(1, 1.56061)$.

To convert the probabilistic variable X back to an estimation $\mathcal{X}$, we would need to know the confidence p'_X, at which we would like to express the estimation. After we have specified p'_X, we can compute the appropriate interval $[k_1, k_2]$ from equation (9). We denote such back-conversion as $X \sim N(a_X, \sigma_X) \to \mathcal{X} = (p'_X, k'_1, k'_2.)$

To simplify the operations with our estimations of the attack-tree node parameters, we will convert all estimations to the same confidence level p_T, which will be defined globally for the attack-tree. In effect, p_T defines the confidence level or the margin of error at which we would like to have the answer of our attack-tree analysis given. If the original estimation $\mathcal{X}$ of an expert is given using some other confidence level p_X, we first convert $\mathcal{X} = (p_X, k_1, k_2) \to X \sim N(a_X, \sigma_X)$ and then find the new interval $[k'_1, k'_2]$ by $X \sim N(a_X, \sigma_X) \to \mathcal{X} = (p_T, k'_1, k'_2.)$

Next, we need to define the computation rules for estimations, i.e. operations $+$ and $\cdot$ as functions of type $\mathbb{P} \times \mathbb{R} \to \mathbb{P}$, operations $+$, $\cdot$ and $/$ as functions of type $\mathbb{P} \times \mathbb{P} \to \mathbb{P}$ and a binary relation $\geqq$ on the set $\mathbb{P}$.

Adding a real number to an estimation. Given $r \in \mathbb{R}$ together with estimation $\mathcal{X} = (p_T, k_1, k_2)$ and wanting to compute $\mathcal{X} + r = \mathcal{Y}$, we first convert $\mathcal{X} \to X \sim N(a_X, \sigma_X)$. After that we can compute $Y = r + X$ using the properties $a_Y = r + a_X$ and $\sigma_Y = \sigma_X$. Finally, we have $Y \to \mathcal{Y} = (p_T, k_1 + r, k_2 + r)$.

Multiplying an estimation by a real number. It is known that given $X \sim N(a_X, \sigma_X)$ and $r \in \mathrm{I\!R}$ we have $r \cdot X = Y \sim N(r \cdot a_X, |r \cdot \sigma_X|)$. To simplify the computation, $\mathcal{X} = (p_T, k_1, k_2)$ is first centralized to $\mathring{\mathcal{X}} = \mathcal{X} - a_X$. Then $\sigma_Y = |r \cdot \sigma_X|$ and now we obtain $\mathring{Y} \to \mathring{\mathcal{Y}} = (p_T, -k, k)$, where the interval $[-k; k]$ is found from the equation

$$p_T = 2\Phi\left(\frac{|k|}{\sigma_X}\right) \quad . \tag{10}$$

After that $\mathring{\mathcal{Y}}$ is de-centralized by shifting its interval by $r \cdot a_X$. Thus, we finally get $\mathcal{Y} = (p_T, -k + r \cdot a_X, k + r \cdot a_X)$.

Adding two estimations $\boldsymbol{\mathcal{X}_1}$ and $\boldsymbol{\mathcal{X}_2}$. When adding two estimations $\mathcal{X}_1 = (p_T, k_1, k_2)$ and $\mathcal{X}_2 = (p_T, k_1', k_2')$, we first convert both of them to $X_1 \sim N(a_1, \sigma_1)$ and $X_2 \sim N(a_2, \sigma_2)$. Then, assuming X_1 and X_2 to be independent, we can compute $Y = X_1 + X_2$, where $a_Y = a_1 + a_2$ and $\sigma_Y = \sigma_1 + \sigma_2$. In case of subtracting, we get $a_Y = a_1 - a_2$, but $\sigma_Y = \sigma_1 + \sigma_2$.

Now we have the necessary information to convert $Y \to \mathcal{Y}$. First we convert $\mathring{Y} \to \mathring{\mathcal{Y}} = (p_T, -k, k)$, where k is found from equation (10) by replacing σ_X with σ_Y. Now, $\mathring{\mathcal{Y}}$ can be de-centralized by shifting its interval by $a_1 + a_2$. So, we get the final result $\mathcal{Y} = (p_T, -k + a_1 + a_2, k + a_1 + a_2)$, or in case of the subtraction, $\mathcal{Y} = (p_T, -k + a_1 - a_2, k + a_1 - a_2)$.

Multiplying two estimations $\boldsymbol{\mathcal{X}_1}$ and $\boldsymbol{\mathcal{X}_2}$. When multiplying two estimations $\mathcal{X}_1$ and $\mathcal{X}_2$ we first centralize them to $\mathring{\mathcal{X}}_1$ and $\mathring{\mathcal{X}}_2$. Then $\mathring{Y} = \mathring{X}_1 \cdot \mathring{X}_2$ is calculated using the assumption of independent variables and the fact that $\sigma_Y = \sigma_1 \cdot \sigma_2$, however, the distribution of $\mathring{Y}$ is not normal, but Bessel distribution.

To convert Y to $\mathcal{Y}$, we compute $\mathring{Y} \to \mathring{\mathcal{Y}} = (p_T, -k_B, k_B)$, where k_B is found from equation

$$p_T = \Pr[-k_B \le Y \le k_B] = \int_{-k_B}^{k_B} f_B(y, \sigma_Y)dy \quad , \tag{11}$$

where

$$f_B(y, \sigma_Y) = \frac{1}{\pi\sigma_Y} K_0\left(\frac{|y|}{\sigma_Y}\right)$$

is the probability density function of the Bessel distribution. Now, $\mathring{\mathcal{Y}}$ needs to be de-centralized. The mean of $\mathcal{Y}$ could be computed as $a_Y = \mathbf{E}(X_1 \cdot X_2) = \mathbf{E}X_1 \cdot \mathbf{E}X_2 = a_1 \cdot a_2$. Therefore, we can shift $\mathring{\mathcal{Y}}$ interval by $a_1 \cdot a_2$. So, we get the final result $\mathcal{Y} = (p_T, -k_B + a_1 \cdot a_2, k_B + a_1 \cdot a_2)$.

Dividing two estimations $\boldsymbol{\mathcal{X}_1}$ and $\boldsymbol{\mathcal{X}_2}$. Using centralized independent variables $\mathring{X}_1$ and $\mathring{X}_2$, it is known that $\mathring{X}_1 / \mathring{X}_2 = \mathring{Y} \sim \mathrm{Cauchy}(0, \sigma_Y)$, where $\sigma_Y = \frac{\sigma_1}{\sigma_2}$.

Using the global confidence value p_T, we convert the Y to $\mathcal{Y}$. In order to do that, we first convert $\mathring{Y} \to \mathring{\mathcal{Y}} = (p_T, -k_C, k_C)$, where k_C is found from the equation

$$p_T = \Pr[-k_B \le Y \le k_B] = \int_{-k_B}^{k_B} f_C(y, y_0, \sigma_Y)dy \quad , \tag{12}$$

where

$$f_C(y, y_0, \sigma_Y) = \frac{\sigma_Y}{\pi} \cdot \frac{1}{\sigma_Y^2 + (y - y_0)^2}$$

is the probability density function of the Cauchy distribution. Now, $\mathring{\mathcal{Y}}$ needs to be decentralized. The mean of $\mathcal{Y}$ can be computed as $a_Y = \mathbf{E}(X_1/X_2) = \mathbf{E}X_1/\mathbf{E}X_2 = a_1/a_2$. Therefore, we can shift $\mathring{\mathcal{Y}}$ interval by a_1/a_2. So, we get the final result $\mathcal{Y} = (p_T, -k_C + a_1/a_2, k_C + a_1/a_2)$.

Comparing two estimations $\mathcal{X}_1$ and $\mathcal{X}_2$. The last operation to enable us to use estimations as operands in our formulae is the comparison. To decide the order of the estimations, we compare the means of the corresponding random variables. Formally, we can define the comparison as the binary relation $\geqq = \{(\mathcal{X}_1; \mathcal{X}_2) | \mathcal{X}_1 = (p_T, k_1, k_2) \to X_1 \sim N(a_1, \sigma_1), \mathcal{X}_2 = (p_T, k_1', k_2') \to X_2 \sim N(a_2, \sigma_2), a_1 \geq a_2\}$.

4.3 Soundness of Computations

Almost all parameters of the nodes have a limited value domain predefined by the interpretation of the parameter, e.g. Cost should be a non-negative real number and p is a probability belonging to the interval $[0, 1]$. When estimations are considered instead of specific values, it is still natural to assume that the respective intervals $[k_1, k_2]$ are subsets of the value domains (e.g. $[k_1, k_2] \subset [0, \infty)$ for Cost and $[k_1, k_2] \subset [0, 1]$ for p). Even if expert estimations given to leaf node parameters satisfy these assumptions, it may happen that as a result of tree computations, some of the parameters in other nodes do not.

Generally, such a situation indicates that no sound conclusions can be drawn on the given confidence level p_T. This problem can be solved in a number of ways.

- The global confidence level p_T can be decreased in order to achieve soundness of estimations in all the nodes. It is possible to find the largest value p_T ensuring sound conclusions and this value can be considered as the confidence level of the whole tree.
- It is possible to define the required confidence level locally for each node.
- It is possible to adjust one or both of the bounds k_1 and k_2 to fit into the required interval; this will automatically decrease the confidence level of the respective node as well.

Each of these approaches has its pros and cons; selecting the best one may be application specific and remains the subject for future research.

5 Tree Computations with Estimations

First consider as an illustration a simple attack tree computation routine based on the example given in Figure 1. First we fix the level of confidence of our estimations to be $p_T = 0.9$ and second we let out experts to evaluate the parameters of the leaves with this confidence. Assume we get the estimation interval for the Cost parameter in the leaf

"Use privileged person to leak information" at this confidence to be $[7.56 \cdot 10^4, 1.24 \cdot 10^5]$, etc, as given in Table 2.

Next we use the computation rules developed in Section 4 to obtain parameter estimations for non-leaf nodes as well (see Table 2). Finally, in the root node we use (6) in its interval form to find $\Pr[2.51 \cdot 10^7 \leq \mathsf{Outcome} \leq 5.36 \cdot 10^7] = 0.9$, which shows that the outcome of the attack is with high probability positive for the attacker, hence some measures must be introduced in order to counter it.

Table 2. Attacktree of leaking sensitive information from a company

ID	**Description**	**Type**	**Parameter estimations**
	Gain of the attacktree		$\Pr[5.23 \cdot 10^7 \leq \mathsf{Gain} \leq 2.48 \cdot 10^8] = 0.9$
A	Leak information	OR	$\Pr[8.58 \cdot 10^4 \leq \mathsf{Cost} \leq 2.14 \cdot 10^5] = 0.9$ $\Pr[0.0561 \leq p \leq 0.544] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_S \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_F \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[2.51 \cdot 10^7 \leq \mathsf{Outcome} \leq 5.36 \cdot 10^7] = 0.9$
A.1	Leak without internal help	OR	$\Pr[8.58 \cdot 10^4 \leq \mathsf{Cost} \leq 2.14 \cdot 10^5] = 0.9$ $\Pr[0.0561 \leq p \leq 0.544] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_S \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_F \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[2.51 \cdot 10^7 \leq \mathsf{Outcome} \leq 5.36 \cdot 10^7] = 0.9$
A.1.1	Electronic espionage	LEAF	$\Pr[9.86 \cdot 10^6 \leq \mathsf{Cost} \leq 1.11 \cdot 10^7] = 0.9$ $\Pr[0.186 \leq p \leq 0.314] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_S \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_F \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[1.71 \cdot 10^7 \leq \mathsf{Outcome} \leq 2.59 \cdot 10^7] = 0.9$
A.1.2	Regular espionage	LEAF	$\Pr[8.58 \cdot 10^4 \leq \mathsf{Cost} \leq 2.14 \cdot 10^5] = 0.9$ $\Pr[0.0561 \leq p \leq 0.544] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_S \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_F \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[2.51 \cdot 10^7 \leq \mathsf{Outcome} \leq 5.36 \cdot 10^7] = 0.9$
A.2	Leak with internal help	AND	$\Pr[8.66 \cdot 10^6 \leq \mathsf{Cost} \leq 2.15 \cdot 10^7] = 0.9$ $\Pr[0.0779 \leq p \leq 0.0871] = 0.9$ $\Pr[9.72 \cdot 10^6 \leq \pi_S \leq 1.23 \cdot 10^7] = 0.9$ $\Pr[-3.33 \cdot 10^6 \leq \pi_F \leq 2.53 \cdot 10^7] = 0.9$ $\Pr[-2.05 \cdot 10^7 \leq \mathsf{Outcome} \leq -6.98 \cdot 10^6] = 0.9$
A.2.1	Recruite privileged person	LEAF	$\Pr[8.58 \cdot 10^6 \leq \mathsf{Cost} \leq 2.14 \cdot 10^7] = 0.9$ $\Pr[0.0858 \leq p \leq 0.214] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_S \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_F \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[-8.16 \cdot 10^6 \leq \mathsf{Outcome} \leq 1.22 \cdot 10^7] = 0.9$
A.2.2	Use privileged person to leak information	LEAF	$\Pr[7.56 \cdot 10^4 \leq \mathsf{Cost} \leq 1.24 \cdot 10^5] = 0.9$ $\Pr[0.428 \leq p \leq 0.672] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_S \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[4.86 \cdot 10^6 \leq \pi_F \leq 6.14 \cdot 10^6] = 0.9$ $\Pr[6.98 \cdot 10^7 \leq \mathsf{Outcome} \leq 8.4 \cdot 10^7] = 0.9$

In general, we may have three possible classes of estimations $\mathcal{X} = (p_X, k_1, k_2)$ for $\mathsf{Outcome}$ of the root node:

1. $0 < k_1 < k_2$, in which case we say that the vulnerability level of the primary threat under consideration with respect to the required confidence level is *high*;
2. $k_1 < k_2 < 0$, in which case we say that the vulnerability level is *low*;
3. $k_1 \leq 0 \leq k_2$, in which case we say that the vulnerability level is *medium*.

If needed, the last class can be further divided into lower medium and higher medium vulnerability levels depending on whether the mean value $\frac{k_1+k_2}{2}$ of the estimation (considered as a normally distributed random variable) is less or greater than zero.

If the security analyst finds out that the security level is not acceptable, (s)he concludes that some measures must be implemented. The possible measures are usually targeted towards lowering attack success probability or increasing expected penalties (e.g. by increasing probability of getting caught). When some set of protection measures is considered, the tree computations can be performed again for a new setting and if the security level becomes acceptable, we know that the set of measures is sufficient. It only remains to pick the most adequate set of such measures (this step is identical to the one described in [16]).

Following the tree computation routine, we can also find out which nodes of the tree are critical ones and must be addressed with our security enhancements. E.g. in the example presented above we see that parameters of the root node are derived from the parameters of the leaf “Regular espionage”, which is thus the most vulnerable node in this setting.

6 Conclusions and Further Work

We presented an extension of the multi-parameter threat tree model to the case where the parameters of elementary attacks are given as interval estimations rather than exact values. Clearly, such a problem setting implies the need to compute the values of primary threat parameters as estimations as well. A suitable method for defining necessary algebraic operations and relations on evaluations was developed in this paper and illustrated with a simple attack scenario.

There are still several directions our research can be continued in. First, our current heuristic used to compute with estimations is to consider all the parameters as normally distributed with suitable parameters. This simplification can turn out to be too restrictive for some applications, hence further studies are needed to find out how other distributions behave under the given tree computation rules.

Second, our tree computation routine can give out-of-bounds values for some parameters (e.g. probabilities) in some nodes. There are several possible solutions to this problem and selecting the best one remains the subject for future research as well.

And, last but not least, even though the authors have used attack tree approach successfully in several security analyses, its extension to interval estimations still needs further practical evaluation.

Acknowledgments

This research has been supported by the Estonian Science Foundation grant no. 7081.

References

1. Parker, D.B.: Fighting Computer Crime: A New Framework for Protecting Information. John Wiley & Sons, Chichester (2001)
2. Geer, D., Hoo, K.S., Jaquith, A.: Information security: Why the future belongs to the quants. IEEE Security and Privacy 1(4), 24–32 (2003)
3. Sonnenreich, W., Albanese, J., Stout, B.: Return On Security Investment (ROSI) – A practical quantitative model. Journal of Research and Practice in Information Technology 38(1), 55–66 (2006)
4. Rieke, R.: Modelling and analysing network security policies in a given vulnerability setting. In: Lopez, J. (ed.) CRITIS 2006. LNCS, vol. 4347, pp. 67–78. Springer, Heidelberg (2006)
5. Sheyner, O., Wing, J.: Tools for generating and analyzing attack graphs. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W-P. (eds.) FMCO 2003. LNCS, vol. 3188, pp. 344–371. Springer, Heidelberg (2004)
6. Meritt, J.W.: A method for quantitative risk analysis. In: Proceedings of the 22nd National Information Systems Security Conference (1999)
7. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated generation and analysis of attack graphs. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA (May 2002)
8. Jha, S., Sheyner, O., Wing, J.: Two formal analyses of attack graphs. In: Proceedings of the 15th IEEE Computer Security Foundations Workshop, pp. 49–63 (2002)
9. Vesely, W., Goldberg, F., Roberts, N., Haasl, D.: Fault Tree Handbook. US Government Printing Office, Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission (January 1981)
10. Viega, J., McGraw, G.: Building Secure Software: How to Avoid Security Problems the Right Way. Addison Wesley Professional, Reading (2001)
11. Moore, A.P., Ellison, R.J., Linger, R.C.: Attack modeling for information security and survivability. Technical Report CMU/SEI-2001-TN-001, Software Engineering Institute (2001)
12. Schneier, B.: Attack trees: Modeling security threats. Dr. Dobb's Journal 24(12), 21–29 (1999)
13. Schneier, B.: Secrets & Lies. Digital Security in a Networked World. John Wiley & Sons, Chichester (2000)
14. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006)
15. Opel, A.: Design and implementation of a support tool for attack trees. Technical report, Otto-von-Guericke University Internship Thesis (March,2005)
16. Buldas, A., Laud, P., Priisalu, J., Saarepera, M., Willemson, J.: Rational Choice of Security Measures via Multi-Parameter Attack Trees. In: Lopez, J. (ed.) CRITIS 2006. LNCS, vol. 4347, pp. 235–248. Springer, Heidelberg (2006)
17. Qin, X., Lee, W.: Attack plan recognition and prediction using causal networks. In: 20th Annual Computer Security Applications Conference, pp. 370–379 (December 2004)
18. Kleiter, G.D.: Propagating imprecise probabilities in bayesian networks. Artificial Intelligence 88(1-2), 143–161 (1996)
19. Borsotto, M., Zhang, W., Kapanci, E., Pfeffer, A., Crick, C.: A junction tree propagation algorithm for bayesian networks with second-order uncertainties. In: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence, pp. 455–464 (2006)

Practical Security Analysis of E-Voting Systems

Ahto Buldas[1,2,3,⋆] and Triinu Mägi[4]

[1] Cybernetica AS. Akadeemia tee 21, 12618 Tallinn, Estonia
[2] Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia
[3] University of Tartu, Liivi 2, 50409 Tartu, Estonia
ahto.buldas@ut.ee
[4] Estonian Ministry of Justice, Center of Registers and Information Systems. Lõkke 4, 19081 Tallinn, Estonia
triinu.magi@just.ee

Abstract. We adapt game theoretic methods for studying the security of two e-voting systems: the Estonian E-Voting System (EstEVS) and Secure Electronic Registration and Voting Experiment (SERVE) performed in the United States of America. While these two systems are quite similar from technical side, security experts have made totally different decisions about their security—EstEVS was indeed used in practical elections while SERVE was decided to be insecure. The aim of this work is to clarify if the minor technical differences between these two systems were indeed a sufficient reason to distinguish between their security. Our analysis is oriented to practical security against large-scale attacks. We define a model for the real-life environment in which voting takes place and analyze the behavior of adversaries. We show that in our model EstEVS is secure and SERVE is not. The reliability of the results is still questionable because of our limited knowledge about many of the parameters. It turns out though that our main results are quite robust with respect to the choice of parameters.

1 Introduction

Many of us have dealt with electronic commerce transactions. This is already a part of everyday life. However, e-voting is not yet so widely used. A secure electronic voting system is still one of the most challenging tasks, because of the need for finding a trade-off between seemingly contradictory requirements like privacy vs. auditability. Thereby, it is difficult to adopt ordinary mechanisms of e-commerce. For example, in e-commerce there is always a possibility to dispute about the content of transactions. Buyers get receipts to prove their participation in transactions. E-voters, in turn, must not get any receipts, because this would enable voters to sell their votes.

In 2003, Estonia initiated the development of an e-voting system (further referred to as Estonian E-Voting System: EstEVS) [12]. The aim was to use e-voting in the elections of the local government councils in 2005. In January 2004, a group of American security experts revealed the security report of Secure Electronic Registration and Voting Experiment (SERVE) [1]. The SERVE system was planned for deployment in the 2004 primary and general elections and allows eligible voters to vote electronically via

[⋆] Partially supported by Estonian SF grant no. 7081, and by EU FP6-15964: "AEOLUS".

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 320–335, 2007.
© Springer-Verlag Berlin Heidelberg 2007

Internet. After examining the security of SERVE, the group of security experts recommended that SERVE should be shut down. They also declared that they do not believe that differently constituted projects could be more secure than SERVE. Their conclusion was that the real barriers to success in e-voting are not skills, resources, etc; it is the fact that given the current Internet and PC security technology, e-voting is an essentially impossible task. The SERVE project was indeed terminated in January 2004. At the same time, Estonia continued to develop an e-voting system and implemented it according to the plans. In their security analysis [2] estonian experts declared that EstEVS is sufficiently secure in practice.

This contradicting situation was the main initiator of this work. From closer view, both security reports are consistent and contain truthful and convincing arguments. One of the main reasons for two totally different decisions was the lack of unified rational security analysis in both reports. Some of the arguments were quite emotional, being based on experts' subjective opinions and ”common wisdom”. The aim of this work is to adapt rational security analysis methods for studying the two e-voting systems. It gives us the possibility to compare practical security levels of these systems.

One of the rational approaches of security is known from theoretical cryptography: security reductions, which are proofs that security conditions held under certain combinatorial assumptions, such as hardness of factoring or Diffie-Hellman problem. For estimating practical security, we also need empirical assumptions about the real world. Moreover, in theoretical cryptography the adversaries are considered to be Turing machines, which are well-defined and relatively easy to study. The real world adversaries are human beings with unpredictable behavior and different motives. Hence, for analyzing practical security, we need models for real world adversaries. In this work, we adapt *multi-parameter attack trees* [3] for analyzing the security of e-voting systems.

Real-world security is not just a technical issue. In many cases, it would be more beneficial for an adversary to bribe employees of organizations rather than to break into their computer system from outside. Hence, the model for real-life environment must consider many ”social parameters” like the costs of bribing people. We create a model for real-life environment in which these parameters are accounted.

We show that EstEVS is practically secure in our model but SERVE has vulnerabilities, which make certain voting-specific attacks possible. Additionally, we show that reasonable changes in the model will not change the results of the analysis. This means that if our environment model indeed reflects the reality, then EstEVS is more secure than SERVE and the security experts' opinions were reasonable. It turns out that the main technical disadvantages of SERVE are: (1) *ballot decryption in e-voting servers*, (2) *lack of independent audit log systems*, (3) *online votes counting server that contains, besides votes, also the identities of voters*, (4) *ballots are not signed by voters*.

We tried to choose the parameters of the model so that they were as close as possible to real society. We used information from Internet, research reports, interviews with public prosecutors and well-studied attack scenarios. In spite of that, our model is obviously not perfect—the estimation of environment characteristics is quite subjective. Still, this work emphasizes the need for better measurements of these environment characteristics, in case we have to analyze the practical security of e-voting systems. Better measurements definitely would improve this security analysis. Unfortunately, it was not possible to include all details of the analysis into this paper. A somewhat more complete representation can be found in the master thesis of Triinu Mägi [13].

2 Security Properties of E-Voting

High security is essential to elections. Democracy relies on broad confidence in the integrity of elections. There has been a lot of attention to electronic voting by cryptographers because of the challenging need to simultaneously achieve many seemingly contradictory properties, like privacy, auditability, and correctness. The most important requirements of e-voting are the following:

i. Eligible voters are able to cast ballots that are counted in the final tally.
ii. Non-eligible voters are disfranchised.
iii. Eligible voters are unable to cast two ballots that are both counted in the final tally.
iv. Voting is private and incoercible. This apparently contradicts correctness, because eligible voters must be identified to distinguish them from non-eligible ones.
v. It is possible for auditors to check whether the final tally is correctly computed. This requirement says that a group of dedicated auditors or Electoral Committee can check the correctness of voting.
vi. The results of voting must be secret until the official end of voting. No one, including votes' counting officers, must be able to reveal the final tally before the official date. Otherwise, the result of voting could affect voters' decisions.

Some researches suggest stronger security properties of e-voting but we concentrate only to the most important properties that directly correspond to the requirements of traditional voting. One of the main starting points of this work is that the security of e-voting should be comparable to that of traditional voting, though we might achieve more by using contemporary cryptographic techniques. The properties listed above are relevant for almost all voting systems and they are the the basis of our security analysis.

For securely implementing e-voting systems in real-life elections cryptographic schemes are clearly not the main problem. A far deeper concern is whether the workstations of "average citizens" (in which computer viruses are everyday visitors) can be used for such a security-critical task.

3 State of the Art

Internet voting systems have been implemented in Europe in couple of places, for example in the Netherlands in 2004 in the European Parliamentary elections. The target group consisted of the Dutch electors' resident abroad and electors resident in the Netherlands who are temporarily abroad on the Election Day. In Great Britain, remote electronic voting systems were used in the local elections of 30 municipalities in 2003.

In the United States of America, many attempts have been made to use e-voting systems. The Voting over the Internet (VOI) project was used in the general elections of 2000 in four states. The Internet votes were legally accepted, but their amount was small (84 votes) [11]. VOI's experiment was too small for being a likely target of attacks.

Another e-voting project named Secure Electronic Registration and Voting Experiment (SERVE) was developed for primary and general elections in 2004 in the United States of America. The eligible voters of SERVE were mainly overseas voters and military personnel. The US Department of Defense terminated SERVE in 2004 because a group of security experts had found that SERVE was not sufficiently secure.

The Estonian e-voting system was applied first time in the municipal elections in 2005. The second implementation was in 2007 in Parliamentary elections. There were 5.4 per cent of e-votes among all votes.

4 Description of E-Voting Systems

In the following, we describe EstEVS and SERVE and emphasize their main differences. The Estonian e-voting system is implemented from the sixth day up to the fourth day before the Election Day. There are two main principles in EstEVS.

(1) Each eligible voter is able to re-vote, so that the older votes are deleted.
(2) Traditional voting cancels electronic votes.

In EstEVS the national Public Key Infrastructure is applied and voters use their authentication and digital signature certificates for casting votes. In SERVE, it is possible to vote any time within 30 days before the Election Day until the closing time of polls on the Election Day. Every voter can vote only once. There are no Public Key Infrastructure and ID-cards used in SERVE. In both e-voting systems if considerable attacks against e-voting have been detected, Electoral Committee might stop e-voting and cancel the result of voting. In general terms, e-voting systems consist into four main components:

- *Voter Applications* - a web application for casting votes.
- *Network Sever* - an server that provides voters an interface for casting their votes.
- *Votes Storing Server* - an server for storing, managing, and maintaining votes.
- *Votes Counting Server* - a server for counting the final tally.

In SERVE, Votes Counting Server is online while in EstEVS it is off-line. Additionally, EstEVS has an independent audit log system which consists of traces of all voting procedures. All log records are cryptographically linked. Log files enable to audit the

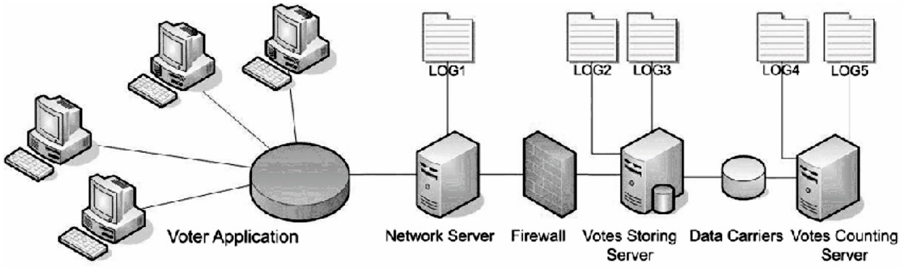


Fig. 1. Components of the Estonian e-voting system

e-voting system. SERVE has a similar architecture to that of EstEVS, except the log files system and the off-line Votes Counting Server.

We now briefly describe the processes of e-voting in EstEVS and SERVE. Fig. 1 depicts the components of the EstEVS. Voting procedure is started with a voter connecting to Network Server via the SSL protocol. Voters enter their personal data for authentication. In EstEVS, national Public Key Infrastructure is applied and voters use their authentication certificates. In SERVE, there is a voters' registration process before the e-voting and voters authenticate themselves with passwords. When the connection is established, then a signed ActiveX control is downloaded to voter's computer in both e-voting systems. An authenticated voter makes his/her choice from a list of candidates transferred from Network Server. In EstEVS the application encrypts the vote by using the public key of Votes Counting Server, however in SERVE the application use the public key of Votes' Storing Server for encrypting the votes.

In SERVE, Voter Application sends an encrypted ballot and voter's personal data to Network Server, which forwards the encrypted ballot and voter's personal data to Votes Storing Server. In EstEVS, voters sign the encrypted ballots with their digital signature certificates. Network Server checks whether the session owner is the same person who signed the encrypted ballot (via ID-card authentication) and in case of positive acknowledgment, transfers the signed and encrypted ballot to Votes Storing Server.

Votes Storing Servers verify voter's franchise and if the voter had already voted. The systems reply to each correctly cast vote with a textual receipt. In EstEVS, after the end of the e-voting period Votes Storing Server cancels multiple ballots and saves the trace of canceled ballots into the log file system. Next, the server separates digital signatures and encrypted ballots. In SERVE, Votes Storing Server *decrypts* the ballots, and separates ballots from personal data. After that, Votes Storing Server encrypts the ballots again without voters' personal data with the public key of Votes Counting Server.

In SERVE, Votes Counting Server downloads the list of voters and the encrypted ballots from Votes Storing Server when Votes Counting Server updates its database. In EstEVS, encrypted ballots are transferred to the off-line Votes Counting Server by using data carriers. For counting votes, Votes Counting Server decrypts the encrypted ballots by using the private key of Votes Counting Server. Only accepted format of votes are counted to the final tally. In EstEVS, Votes Counting Server outputs the final tally and in the SERVE system it outputs the final tally and the list of voters. Table 1 depicts the main differences between the two systems.

Table 1. Differences between the two e-voting systems

Characteristic	**EstEVS**	**SERVE**
e-voting used on the Election Day	No	Yes
Possibility to re-vote at the polling station	Yes	No
National Public Key Infrastructure	Yes	No
Voters sign the ballots	Yes	No
State of votes in Votes Storing Server	Encrypted	Not encrypted
State of Votes Counting Server	Off-line	On-line
Audit log system	Yes	No

5 Analysis Method

To measure the real security of e-voting, we should analyze the security in an objective way. It would be insufficient (at least for the purposes of this work) to rely on subjective opinions of security experts—we try to put their opinions to solid ground by providing them with a method to determine whether the system is secure.

In order to declare that e-voting system is secure it must be as secure as traditional voting, which is considered to be practically secure and resistant to *large-scale threats*. This means that the e-voting systems must also be secure against the large-scale voting-specific attacks. A large-scale attack may cause considerable changes in the final tally or reveal large number of votes. Therefore, for estimating practical security of the systems, we try to create an environment model as close as possible to the real-life environment in which e-voting systems are used. In addition to technological parameters, we have to make assumptions about society, people, and motives of attackers. We assume that adversaries are gain-oriented and attack on purpose—to affect the result of elections. We analyze adversarial behavior by using the game-theoretical setting suggested in [3]. According to this setting, attacks are viewed as games the profitability of which (for attackers) depends on the following parameters of the environment model:

- Gains - the gains of the attacker, in case the attack succeeds.
- Costs - the cost of the attack.
- p - the success probability of the attack.
- q - the probability of getting caught (if the attack was successful).
- $\mathsf{Penalties}$ - the penalties in case the attacker is caught (if the attack was successful).
- q_- - the probability of getting caught (in case the attack was not successful).
- $\mathsf{Penalties}_-$ - penalties if the attacker was caught and the attack was unsuccessful.
- $\mathsf{Outcome}$ - average outcome of an attacker.

Considering all these parameters, rational attackers calculate the expected outcome of the game, which determines their decision about whether to attack or not:

$$\mathsf{Outcome} = -\mathsf{Costs} + p \cdot (\mathsf{Gains} - q \cdot \mathsf{Penalties}) - (1 - p) \cdot q_- \cdot \mathsf{Penalties}_- \quad .$$

Attackers do not attack, if the outcome of the attack-game is negative and they always choose the most profitable ways for attacking. For the sake of simplicity we denote:

- by π the average penalty if the attack was successful, i.e. $\pi = q \cdot \mathsf{Penalties}$;
- by π_- the average penalty if the attack was unsuccessful, i.e. the outcome is equal to $-\mathsf{Costs} + p \cdot (\mathsf{Gains} - \pi) - (1 - p) \cdot \pi_-$.

For better estimation of the parameters, attacks are split into simpler ones by two rules. *AND-rule* states that the component-attacks are all necessary for the original attack, whereas *OR-rule* states that at least one of the components is needed for the original attack. Such a decomposition procedure is iterated until we can estimate the parameters of all components, i.e. they can be deduced from our model of environment. The composition tree that corresponds to this process is called an *attack tree* [5]. Each node of an attack tree represents an attack. Leaf nodes represent atomic attacks for which all parameters are known. For simplicity, it is assumed [3] that Gains is the same for all nodes. To compute the parameters of the root node, we need the following rules [3]:

- For an OR-node with children $(\mathsf{Costs}_1, p_1, \pi_1, \pi_{1-})$ and $(\mathsf{Costs}_2, p_2, \pi_2, \pi_{2-})$ the parameters $(\mathsf{Costs}, p, \pi, \pi_-)$ are computed as follows:

$$(\mathsf{Costs}, p, \pi, \pi_-) = \begin{cases} (\mathsf{Costs}_1, p_1, \pi_1, \pi_{1-}), \text{ if } \mathsf{Outcome}_1 > \mathsf{Outcome}_2 \\ (\mathsf{Costs}_2, p_2, \pi_2, \pi_{2-}), \text{ if } \mathsf{Outcome}_1 \leq \mathsf{Outcome}_2 \end{cases},$$

  where $\mathsf{Outcome}_i = -\mathsf{Costs}_i + p_i \cdot \mathsf{Gains} - p_i \cdot \pi_i - (1 - p_i) \cdot \pi_{i-}$ for $i = 1, 2$.
- For a AND-node with children $(\mathsf{Costs}_1, p_1, \pi_1, \pi_{1-})$ and $(\mathsf{Costs}_2, p_2, \pi_2, \pi_{2-})$ the parameters $(\mathsf{Costs}, p, \pi, \pi_-)$ are defined as follows (where $\tilde{p_i}$ will denote $1 - p_i$):

$$\mathsf{Costs} = \mathsf{Costs}_1 + \mathsf{Costs}_2; \qquad p = p_1 \cdot p_2; \qquad \pi = \pi_1 + \pi_2; \quad \text{and}$$
$$\pi_- = \frac{p_1 \tilde{p_2}(\pi_1 + \pi_{2-}) + \tilde{p_1} p_2 (\pi_{1-} + \pi_2) + \tilde{p_1}\tilde{p_2}(\pi_{1-} + \pi_{2-})}{1 - p_1 p_2}.$$

6 Adversarial Model and Threats

In our analysis, we consider adversaries as a rationally thinking persons who always choose the most profitable attacks and who will not attack if all possible attacks are unprofitable. We do not model adversaries as inside attackers. We assume that the development team of e-voting has been created carefully and the team members are benevolent by themselves. However, we assume that the team members can be influenced from outside (for example, bribing) in order to affect an e-voting system maliciously. In this work, we do not analyze crimes against person because they pose an equal threat to traditional voting. Hence, we do not consider that anybody is involved in an attack by coercion or violence. We analyze the behavior of attackers through the components of e-voting systems by using multi-parameter attack trees [3]. For example an adversary has the following activities:

- to attack Voter Application in order to affect the votes' casting process;
- to attack the connection between Voter Application and Network Server in order to affect the votes before they are received by Network Server;
- to attack Network Server in order to affect the votes' reception.

By using the activities that are allowed in the adversarial model, the aim of attacker is to perform a large-scale attack. Small-scale attacks, which affect a small number of votes, do not affect the overall result of voting and hence do not pose a threat to democratic society as a whole. So, we will study large-scale attacks that cause considerable changes in the final tally or a large scale of votes to become revealed.

How big is *large number*? How many votes should be changed or revealed in e-voting systems so that we may talk about a large-scale attack? For estimating this parameter we analyzed elections in Estonia and in the United States of America. We saw that the minimum average per cent of votes to affect the result of voting could be 4 per cent [4,9]. There has been an exception in the presidential elections of the USA in 2004. The difference between the rates of parties was only 0.0246. The number of target voters of EstEVS and of SERVE was 1 million and 6 millions, respectively. Obviously, one hundred infected computers do not affect the overall result of elections. If 1,000 computers are infected, it would be possible to affect 0.1 per cent of the Estonian votes and 0.016 per cent of the United States votes. To summarize, we consider that infecting 1,000 computers is sufficient for having a large-scale attack in e-voting systems.

We say that e-voting is *practically secure* if it resists the following large-scale attacks:

1) *Large-scale votes' theft.* The aim of the attack is to change votes or to give more (forged) votes for favorite candidates. Such an attack is possible only if the adversary is able to cast ballots in the name of many users or the system enables voters to cast multiple ballots that are all counted in the final tally.
2) *Large-scale disfranchisement of votes.* It means that a large number of correctly encrypted ballots from eligible voters never reach Votes Storing Server. Attack could also selectively disfranchise eligible votes in order to eliminate undesirable ones. Note that the aim of a rational (and well-prepared) attacker is not to cause the overall failure of e-voting and hence such an attack should stay unnoticed.
3) *Large-scale votes' buying and selling.* It means that a large number of votes are sold. The aim of the attack is to increase the amount of votes for certain candidates.
4) *Large-scale privacy violation.* The aim of the attack is to reveal how voters have voted. This my cause violence and persecution in the society.

7 Security Assumptions

We make several simplifying assumptions which will eliminate many irrelevant details of our empirical analysis and thereby keeps the "big picture" of the analysis observable to the reader. For example, we assume that the cryptographic schemes used in e-voting systems are secure. Our analysis uses the following assumptions:

- *Assumption I*: It is impossible to forge signatures without private keys.
- *Assumption II*: It is impossible to deduce votes from encrypted ballots.
- *Assumption III*: Adversaries do not have access to the private keys of voting servers. Key management at the server side is sufficient to prevent key compromise.
- *Assumption IV*: Voters' registration is secure. EstEVS uses national PKI and does not need voters' registration—ID-cards with authentication and digital signature certificates are issued to all citizens. In this work, we assume that sharing of authentication data and digital signature certification is secure in EstEVS. For fair comparison of the two systems, we also assume that the phase of voter's registration in SERVE is secure.
- *Assumption V*: The phase of votes' counting behaves as specified. All correctly cast votes that are received by Votes Counting Server are counted correctly. This assumption might be unjustified in voting systems, because the insider threats are even more common than the outsider threats. However, in this analysis the insider threats of votes' counting phase are not taken into account.
- *Assumption VI*: The log file system of EstEVS is secure. All records in audit logs are cryptographically linked and it is impossible to modify them without detection.
- *Assumption VII*: If considerable attacks are detected that cause misbehavior of e-voting then e-voting is immediately stopped and the results of e-voting canceled. Both EstEVS and the SERVE project have justified this property in the requirements of the systems. This is decided by court or Electoral Committee.

8 Model of Environment

A meaningful comparison of two systems must be based on equal or comparable benchmarks. Hence, we create the same environment for the both e-voting systems. It is clear, that the environments of EstEVS and SERVE are different in real life. Moreover, it is even hard to describe these environments adequately and give real characteristics of environment. For example, it is hard to estimate what is the probability of catching and convicting attackers, if voters deliberately create connections to an actively compromised voting server. For adequately specifying the characteristics of an environment for e-voting systems, it is necessary to study the motives and purposes of attacks, success probabilities of attacks, detection probabilities of attacks, awareness of computers' users, punishments for cyber-crimes, etc. In order to make rational decisions about practical security of e-voting systems, we have to know these parameters with sufficient accuracy. Note that if we are unable to do it, then this would also mean that *we do not know whether these systems are secure*. Hence, the way to go here is to obtain better estimates for these parameters.

We create a hypothetical environment for analyzing security of the two e-voting systems. We try to estimate the parameters of the environment as close as possible to the real society. For estimating these parameters we have used information from Internet, from research papers, interviews with specialists and typical attacking scenarios. We assume that typical attackers do not make extensive social research for getting information on whether it is profitable to attack. Quite probably, gain-oriented attackers would analyze the same information from Internet and make decisions intuitively. Definitely, this hypothetical environment is not perfect, but it is the best we can do for comparing the security of the two e-voting systems. Our model contains assumptions about: (1) society, (2) people, (3) technical vulnerabilities, and (4) detection. These assumptions are commented in the following subsections.

8.1 Assumptions About Society

Voting is a fundamental tool of democracy and one of the main rights in democratic society. We assume that the environment we model is a well-developed democratic society in which *the aim of crime determines the seriousness of crime*. If the aim of the crime is to affect the result of voting then it is viewed as a serious crime against society, no matter how it was performed and whether the crime was ”technically successful”. Hence, we assume that $\mathsf{Penalties} \approx \mathsf{Penalties}_{-}$. Moreover, the punishment for crime is at least dispossession of the gains obtained from the crime. Thus, we assume that $\mathsf{Gains} \leq \mathsf{Penalties}$. For simplicity, we study the limit case $\mathsf{Penalties} \approx \mathsf{Gains} \approx \mathsf{Penalties}_{-}$, which implies

$$\mathsf{Outcome} \approx -\mathsf{Costs} + \mathsf{Gains} \cdot [p \cdot (1-q) - (1-p) \cdot q_{-}] \enspace .$$

Parties spend lots of money for campaigns of election. Probably, the gain is even bigger. In Estonia, parties spend about $2 million [10] for a campaign of election. We assume that Gains of affecting the result of election is at least 5 times bigger, so $10 million.

By the data available in Internet the price of obtaining malicious code is about $50. A person can be bribed for about $50,000 [7]. We assume that attackers are rationally and economically thinking. Hence, to calculate the cost of attack, we focus on self-cost. Even, if the price of developing a forged Network Server is \$2 million, the expenses of attacks are small compared to the gains.

Considering the specificity of elections, Costs are always much smaller than Gains. Hence, the value of Costs does not affect attacker's final decision to attack an e-voting system or not. Therefore, we may even assume that $\mathsf{Costs} \approx 0$. If the e-voting system is secure when $\mathsf{Costs} = 0$ then the system is also secure when $\mathsf{Costs} > 0$. Therefore, under these simplification we conclude that

$$\mathsf{Outcome} \approx -\mathsf{Costs} + \mathsf{Gains} \cdot [p \cdot (1-q) - (1-p) \cdot q_-] < 0 \ ,$$

whenever $p \cdot (1-q) - (1-p) \cdot q_- < 0$. To summarize, considering the particularity of e-voting we may estimate only three parameters p, q and q_- of the attack game for estimating the profitability of attacks. In the following we list the characteristic probabilities (Char. 1-15) of the environment that we use in our analysis:

8.2 Characteristic Probabilities

In the following, we list 15 characteristic probabilities of the environment that we use in our security analysis. These probabilities are divided into assumptions about: (1) people (Char. 1-7), (2) technical vulnerabilities (Char. 8-11), and (3) detection (Char. 12-15).

Char. 1. About 1 per cent of voters will notice that their computers are infected and will inform the authorities about it. Thereby, the success probability of attacking large number (1000) of voters' workstations (without this being noticed) is $p \leq 0.99^{1000}$.

Char. 2. At least 1 per cent of electronic voters verify the authenticity of the Network Server certificate, the signature of ActiveX component and wait for the confirmation of e-voting. We assume that if a voter is aware of the need to verify the certificate of Network Server, then he is also aware of the need to verify the signature of ActiveX component and to wait for the confirmation about accepted vote. The probability that 1,000 voters do not verify the certificate of Network Server or the signature of ActiveX component or do not wait for the signed confirmation from the e-voting server is $p \leq 0.99^{1000}$. Such a modeling of voters is somewhat idealistic, because all voters are assumed to have the same values of probability. In practice, the attacker may estimate these values by guessing the technical skills and carefulness of the voters and then to attack those with lower skill and careless.

Char. 3. About 33 per cent of people can be bribed for \$50,000 [7].

Char. 4. The probability that voters click on a (well-created) malicious link is ≈ 0.6. Hence, the probability that a fixed set of 1000 people will use the link is $p \leq 0.6^{1000}$.

Char. 5. About 1 per cent of people involved in attacks will reveal information that causes the attackers to be caught. Hence, the probability that a group of 10 people will get caught is $q \geq 1 - 0.99^{10} \approx 0.096$.

Char. 6. We assume intuitively that voter would sell his vote with probability 0.5 by using active votes' selling environment. The probability that voter would sell the vote by using more anonymous ways is 0.7. It means that a voter would feel more secure to participate in a scheme of votes' selling and buying by using computer based voting data saving and proving software.

Char. 7. The probability that voters agree to vote many times (for an attacker) is 0.9.

Char. 8. The probability of exploiting a bug in an operating system or hardware and getting access to a system is ≈ 0.002. We assume that bugs in operating systems or in hardware are discovered once in 3 years on average. Within 2 days, viruses can exploit the bug. Within 7 days, there will be countermeasures available. Hence, attackers have one week per three years to exploit the bug. Thereby, at every moment, there is bug to exploit with probability 0.0064. The probability of getting unauthorized access to administrative areas of a system or to other internal modules is 0.21 [8]. Hence, the probability of exploiting a bug and getting access to the system is $0.0064 \cdot 0.21 \leq 0.002$.

Char. 9. The probability that a forged Network Server or malicious code succeeds in attack is $p \approx 0.95$. Usually, the accordance between functions of developed information system and claimed system requirements is not 95 per cent. However, for estimating the security of system we promote attackers. If the system is secure against powerful and penetrating attacks, then it is secure against weaker attacks.

Char. 10. The probability that voters' computers are vulnerable is about 0.31 [6].

Char. 11. The probability that adversaries have succeeded to gain control over the connection between the e-voting servers is 0.15. We assume intuitively that if the probability that voters' computers are vulnerable for session controlling is 0.31 [6], then the control over the session between servers is harder at least twice as hard, i.e. $p \leq 0.15$.

Char. 12. Code review and auditing can detect about 30% of software errors.

Char. 13. Bribing that causes damage is detected with probability $q \leq 0.3$ [7].

Char. 14. Attacks against insecure server conf. are detected with probability 0.05.

Char. 15. The probability that a successful crime against the e-voting system will be convicted is 0.8. Unsuccessful crimes will be convicted with probability 0.2 [7].

9 Attack Game Analysis

First, we decompose the four large-scale attacks (listed in Sec. 6) by using the OR-rule, i.e. we created a list of alternative ways of attacking the two e-voting systems. After that, we analyzed all alternatives separately by using the security assumptions (Sec. 7), the environment characteristics (Sec. 8), and the attack tree method (Sec. 5). Table 2 depicts the decomposition of the large-scale attacks.[1] For example, we studied seven alternatives for large-scale votes' theft and four alternatives (so called *sub-attacks*) for large-scale disfranchisement attack. For the lack of space, we will not present detailed analysis of all possible attacks and alternatives. First, we focus on a few sub-attacks and then present a more complete analysis for the *large-scale votes' buying attack*.

[1] The table is not complete and does not contain decomposition of all four large scale attacks.

Table 2. Sub attacks for large-scale voting specific attacks. By '-' we mean that the sub-attack is impossible or insufficient.

Attack	Sub-attacks	EstEVS	SERVE
Large-scale votes' theft	Large-scale control over voters' processes	unprofitable	unprofitable
	Large-scale access to voters' private keys	unprofitable	unprofitable
	Eligible voters cast votes more than once	unprofitable	unprofitable
	Large-scale disfranchisement in two servers	unprofitable	-
	Large-scale modification of ballots in the connection between Voter Application and Network Server	-	unprofitable
	Control over processes of Votes Storing Server	-	profitable
	Large-scale votes' adding in Votes Counting Server	-	unprofitable
Large-scale disfranchisement of votes	Large-scale control over voter processes	unprofitable	unprofitable
	Large-scale disfr. before receiving votes	unprofitable	unprofitable
	Large-scale disfr. in two servers	unprofitable	unprofitable
	Control over processes of Votes Storing Server	-	profitable
Large-scale votes' buying/selling	(decomposition omitted)	unprofitable	profitable
Large-scale privacy violation	(decomposition omitted)	unprofitable	profitable

Large-scale control over voters' processes. In EstEVS and in SERVE, large-scale control over voters' processes is possible either by infecting computers one-by-one or by using automatically propagating attacking software (viruses etc.). We assume that both methods have the same expenses. By assumptions, with probability $p \leq 0.99^{1000}$ attackers are able to smuggle malicious code into voters' computers and get the desired data by Char. 1. A large-scale access to voters' private keys is a serious attack and the estimation of detecting the attack is 0.8 by Char. 15. If we assume that the attack was not successful, then the probability of getting caught is $q \geq 0.096$ by Char. 5. For estimating the profitability, we compute $\mathsf{Outcome}$ as follows:

$$\begin{aligned}\mathsf{Outcome} &\leq -\mathsf{Costs}+\mathsf{Gains}\cdot[p\cdot(1-q)-(1-p)\cdot q_{-}]\\ &= -\mathsf{Costs}+\mathsf{Gains}\cdot[0.99^{1000}\cdot(1-0.8)-(1-0.99^{1000})\cdot 0.096]\\ &< -\mathsf{Costs}-\mathsf{Gains}\cdot 0.096 < 0\ .\end{aligned}$$

As $\mathsf{Gains} \gg \mathsf{Costs}$, the value of Costs does not affect the attacker's final decision. The attack is unprofitable, if $p(1-q)-(1-p)q_{-} < 0$. Additionally, even if the probabilities q and q_{-} of getting caught are 0.096, the attack is not profitable. Therefore, an attack via large-scale control over the voters' processes is unprofitable in both systems.

Large-scale access to voters' private keys. An average voter is unable to keep its own workstation secure enough to exclude all possible abuses of the private key. For example, adversaries can steal voter's password for activating the ID-card. Still, it is not possible to arrange a large-scale theft of cards, because voters would notice it immediately and elections will be canceled by Assumption VII. The success probability of large-scale access to voters' private keys is $p \leq 0.99^{1000}$ by Char. 1. The argumentations used here are similar to the previously analyzed large-scale attack. Therefore, large-scale access to the voters' private keys is unprofitable for rational attackers in both e-voting systems.

Large scale votes' buying. Large-scale buying and selling of votes is possible only if there is a possibility to prove a vote. In case the voter could not prove how he/she had voted, the votes' buying and selling is not a trustful deal. There is a theoretical advantage for adversaries in the e-voting systems compared to adversaries in traditional voting. The adversaries do not have to physically contact with every voter for affecting his choice. The adversaries should affect at least 1,000 voters for affecting the result of e-voting. Obviously, the easiest way to affect many voters is to offer votes' buying and selling services. In Section 8, we assumed that Gains of an attack could be $10 million. Let us analyze, whether it is possible to eliminate the parameter Costs like we did previously. Obviously, the price of organizing and preparing the attack is much smaller than Gains. The biggest expense is the price of votes. In the case when adversaries spend 20 per cent of the profit for buying 1,000 votes, the price of vote is $2,000. We assume that such price is attractive for vote sellers. Therefore, Costs for buying at least 1,000 votes is smaller than Gains. In the following, we create attack trees for large scale votes' buying in SERVE and in EstEVS.

9.1 Analysis of SERVE

An attack tree for large-scale votes' buying in SERVE is depicted in Fig. 2 (left) and the computations in Table 3. There are three possibilities to arrange votes' buying and selling in SERVE. First, by using votes selling and buying web server (Sub tree A). Voters connect to votes buying server for casting their votes. The server saves voters' choices and sends ballots to Network Server. Second, voters use votes saving software for getting the receipt of voting and cast a vote directly to Network Server (Sub tree B). A receipt consists of voter's data, a vote, a random number and an encrypted ballot. The voters send the receipts to the adversary for proving how they voted. The adversary attacks the e-voting server for getting a proof that a ballot is received. For inserting malicious code into servers there are four possibilities: software developer of a server is bribed (B.3.2.1.), server administrator is bribed (B.3.2.2.), insecure configuration management is exploited (B.3.2.3.). Third, the adversary attacks the servers of e-voting for checking how voters voted (Sub tree C). Votes Storing Server of SERVE decrypts the ballots. Adversary attacks against Votes Storing Server for the purpose of stealing pairs of voters' data and ballots. These pairs give a proof how voters voted. In the following, we analyze these sub-trees.

Sub-tree A: With probability 0.95 votes' buying and selling information system is developed successfully by Char. 9. To consider the active and public attack, the probability of detecting the attacking group is 0.8 by Char. 15. For analyzing voters connection to votes' buying server, we assume that that 50 per cent of voters would sell their vote by Char. 6. The probability of detecting voters who have voted by using votes buying server is 0.8 because this is the probability of detecting the votes buying server. To summarize, it is not profitable for attacker to attack through votes' buying and selling server.

Sub-tree B: The probability of the votes' saving software functioning correctly is 0.95 by Char. 9. The probability of detecting the votes' saving software is 0.096 by Char. 5. The success probability of voters using the software is $p = 0.7$ by Char. 6. If there are at least 1,000 people involved and Char. 5 is justified then the probability of detection and punishment of saving the receipt is $q = q_- = 1 - 0.99^{1000}$. The probability of the malicious code successfully getting voters' data and encrypted ballots from a voting server is 0.95 by Char. 9. The detection probability is 0.096 by Char. 5. According to Char. 3, a software developer and a server administrator are bribed with probability 0.33. Based on the assumption that development teams use code reviews, misbehavior in software is detected with probability 0.3 by Char. 12. Therefore, for estimating the probability of a software developer getting caught, we consider information leaking and the detection rate of misbehavior in server. Hence, the probability of getting caught without succeeding is $q_- = 0.096 + 0.3 = 0.396$ by Char. 5 and Char. 12. Bribery is detected with probability 0.3 by Char. 13. The success probability of detecting a software developer is $q \approx 0.096 + 0.3 + 0.3 \approx 0.7$ by Char. 5, Char. 12 and Char. 13. In the event that the attack was not successful, the probability of detecting that the server administrator was bribed is at least $q_- \geq 0.096$ by Char. 5. Considering the value of q_- and Char. 13, the probability of a server administrator being caught is $q \approx 0.096 + 0.3 \approx 0.4$. Insecure configuration management is successfully exploited with probability $p \leq 0.002$ by Char. 8. We assume intuitively that the probability of detection of the exploiting configuration management is 0.05 by Char. 14. Control over the connection between servers is successful with probability 0.15 and the probability of the detection of the attack is 0.096 by Char. 11 and Char. 5. To summarize, spreading votes' receipt software does not give a profitable attack in our model.

Sub tree C: The analysis of sub-tree is analogous to the analysis of sub-tree B.3. Attacking Votes Storing Server for getting voters' ballots is successful with probability $p \approx 0.32$ and it has positive Outcome of the attack game. Therefore, Large-scale votes' buying in SERVE is profitable, considering our model.

9.2 Analysis of EstEVS

The attack tree Large-scale votes' buying for EstEVS is depicted in Fig. 2 (right) and the computations in Table 4. Votes buying attack against EstEVS has just one option.

Table 3. Large-scale votes' buying in SERVE

Node	Description of attack	Type	p	q	q_-	π	π_-	Outcome
A	Votes buying server.	AND	0.475	0.64	0.64	$1.6 \cdot 10^7$	$8.7 \cdot 10^6$	$-7.45 \cdot 10^6$
A.1	Attacking software is developed.		0.95	0.8	0.8	$8.0 \cdot 10^6$	$8.0 \cdot 10^6$	$1.5 \cdot 10^6$
A.2	Voters connect to the server.		0.5	0.8	0.8	$8.0 \cdot 10^6$	$8.0 \cdot 10^6$	$-3.0 \cdot 10^6$
B	Spreading votes' receipt software.	AND	0.208	0.0037	0.00089	$1.59 \cdot 10^7$	$1.42 \cdot 10^7$	$-1.23 \cdot 10^7$
B.1	Developing data-saving software.		0.95	0.096	0.096	$9.6 \cdot 10^5$	$9.6 \cdot 10^5$	$8.54 \cdot 10^6$
B.2	Voters use software to save receipts.		0.7	0.999957	0.999957	$1.0 \cdot 10^7$	$1.0 \cdot 10^7$	$-3.0 \cdot 10^6$
B.3	Obtain ballots from server.	AND	0.3135	0.0384	0.0092	$4.96 \cdot 10^6$	$1.40 \cdot 10^6$	$6.21 \cdot 10^5$
B.3.1	Developing malicious code		0.95	0.096	0.096	$9.6 \cdot 10^5$	$9.6 \cdot 10^5$	$8.54 \cdot 10^6$
B.3.2	Inserting code into server.	OR	0.33	0.4	0.096	$4.0 \cdot 10^6$	$9.6 \cdot 10^5$	$1.34 \cdot 10^6$
B.3.2.1	Software developer is bribed.		0.33	0.7	0.396	$7.0 \cdot 10^6$	$3.96 \cdot 10^6$	$-1.6 \cdot 10^6$
B.3.2.2	Server administrator is bribed.		0.33	0.4	0.096	$4 \cdot 10^6$	$9.6 \cdot 10^5$	$1.34 \cdot 10^6$
B.3.2.3	Insecure configuration is exploited.		0.002	0.05	0.05	$5.0 \cdot 10^5$	$5.0 \cdot 10^5$	$-4.8 \cdot 10^5$
B.3.2.4	Control connections between servers.		0.15	0.096	0.096	$9.6 \cdot 10^5$	$9.6 \cdot 10^5$	$5.4 \cdot 10^5$
C	Get ballots from Votes Storing Server.	AND	0.3135	0.0384	0.0092	$4.96 \cdot 10^6$	$1.40 \cdot 10^6$	$6.21 \cdot 10^5$
C.1	Develop malicious vote-saving code.		0.95	0.096	0.096	$9.6 \cdot 10^5$	$9.6 \cdot 10^5$	$8.54 \cdot 10^6$
C.2	Inserting code into server.	OR	0.33	0.4	0.096	$4.0 \cdot 10^6$	$9.6 \cdot 10^5$	$1.34 \cdot 10^6$
C.2.1	Software developer is bribed.		0.33	0.7	0.396	$7.0 \cdot 10^6$	$3.96 \cdot 10^6$	$-1.6 \cdot 10^6$
C.2.2	Server administrator is bribed.		0.33	0.4	0.096	$4 \cdot 10^6$	$9.6 \cdot 10^5$	$1.34 \cdot 10^6$
C.2.3	Insecure configuration is exploited.		0.002	0.05	0.1	$5.0 \cdot 10^5$	$1.0 \cdot 10^6$	$-9.79 \cdot 10^5$

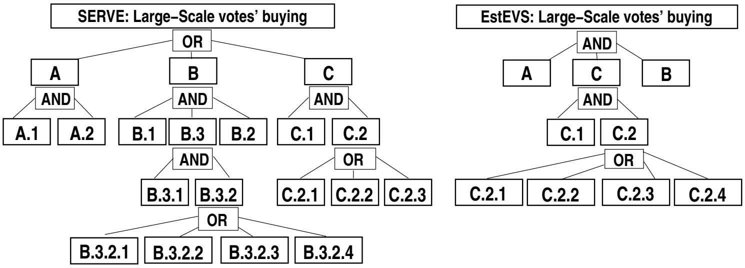


Fig. 2. Attack trees for large-scale votes' buying attack in SERVE (left) and in EstEVS (right)

Adversaries develop software for saving voting data as a receipt. Voters who wish to sell their votes use the software in their computers for delivering the voting receipt. Adversaries attack an e-voting server for getting pairs of voters' data and encrypted ballots. The comparison of receipts and encrypted ballots gives the proof how voters had voted. Without getting control over one of the voting servers it would not be sure whether the ballots were really sent to the voting server.

In EstEVS, it would be impossible to sell votes via votes' buying server because Network Server verifies if the session owner is the same person who signed the ballot. Votes' buying server cannot impersonate voters without having access to their ID-cards.

In EstEVS, the ballots stay encrypted until the votes' counting phase. Hence, without the private key of Votes Counting Server it is insufficient to attack the voting server for checking how voters voted. By Assumption III, adversaries do not have the key.

The tree for votes' buying in EstEVS is similar to the sub-tree B of the corresponding tree of SERVE. Outcome of this tree is negative and hence EstEVS is secure against large-scale votes' buying in our model.

Modification of environment characteristics. For getting some more justification to our choices of parameters, we made some non-systematic robustness tests. We tried to change the environment characteristics so that the value of attack game would change its sign (from positive to negative, or vice versa). For example, we had to decrease Char 2. about 10 times for inverting a value of an attack game. It turned out that several parameters had to be changed simultaneously for inverting game values and keeping

Table 4. Large-scale votes' buying in EstEVS

Node	Description of attack	Type	p	q	q_-	π	π_-	Outcome
	Large-scale votes' buying.	AND	0.2085			$1.59 \cdot 10^7$	$1.42 \cdot 10^7$	-1.2510^7
A	Develop data-saving software.		0.95	0.096	0.096	$9.6 \cdot 10^5$	$9.6 \cdot 10^5$	$8.54 \cdot 10^6$
B	Voters use the software.		0.7	0.999957	0.999957	$1.0 \cdot 10^7$	$1.0 \cdot 10^7$	$-3.0 \cdot 10^6$
C	Obtain ballots from voting server.	AND	0.3135	0.0384	0.0092	$4.96 \cdot 10^6$	$1.4 \cdot 10^6$	$6.21 \cdot 10^5$
C.1	Develop malicious code.		0.95	0.096	0.096	$9.6 \cdot 10^5$	$9.6 \cdot 10^5$	$8.54 \cdot 10^6$
C.2	Insert the code into server.	OR	0.33	0.4	0.096	$4.0 \cdot 10^6$	$9.6 \cdot 10^5$	$1.34 \cdot 10^6$
C.2.1	Software developer is bribed.		0.33	0.7	0.396	$7.0 \cdot 10^6$	$3.96 \cdot 10^6$	$-1.66 \cdot 10^6$
C.2.2	Server administrator is bribed.		0.33	0.4	0.096	$4.0\dot{1}0^6$	$9.6 \cdot 10^5$	$1.34 \cdot 10^6$
C.2.3	Insecure configuration is exploited.		0.002	0.05	0.05	$5.0 \cdot 10^5$	$5.0 \cdot 10^5$	$-4.8 \cdot 10^5$
C.2.4	Control the connection between servers.		0.15	0.096	0.096	$9.6 \cdot 10^5$	$9.6 \cdot 10^5$	$5.4 \cdot 10^5$

changes reasonable (10 times is clearly too much!). Also the punishment and detection characteristics had to be changed approximately 10 times in order to invert game values. So, it turned out that reasonable changes do not have much influence on the final results of our analysis, which to some extent increases our belief about the truthfulness of the results. However, limited knowledge about the real values and the embryonic state of the robustness analysis do not enable to make any conclusions about the real security of these two systems.

10 Further Work

The results of the work are still disputable and need further improvement and justifications, because the characteristics of the defined environment model are arguable. However, this work is one of the first attempts to rationally analyze the security of e-voting by combining both the technical and the social aspects, which are all necessary for making any reliable decisions about the real security of e-voting systems (i.e. when they are applied in real elections in a real society). It is therefore necessary to continue the study about society characteristics for creating more realistic environment models.

References

1. Jefferson, D., Rubin, A.D., Simons, B., Wagner, D.:A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)(2004)
2. Ansper, A., Buldas, A., Oruaas, M., Priisalu, J., Veldre, A., Willemson, J., Kivinurm, K.: The Security of Conception of E-voting: Analysis and Measures. E-hääletamise kontseptsiooni turve: analüüs ja meetmed (2003)
3. Buldas, A., Laud, P., Priisalu, J., Saarepera, M., Willemson, J.: Rational Choice of Security Measures via Multi-Parameter Attack Trees. In: Lopez, J. (ed.) CRITIS 2006. LNCS, vol. 4347, pp. 235–248. Springer, Heidelberg (2006)
4. Estonian National Electoral Committee home page, http://www.vvk.ee
5. Schneier, B.: Attack Trees. Dr. Dobb's Journal, Modeling Security Threats (December 1999)
6. Geer, D., Hoo, K.S., Jaquith, A.: Information Security: Why the Future Belongs to the Quants? IEEE Security and Privacy, 32–40 (July/August 2003)
7. Interview with Estonian public prosecutor Mr. Margus Kurm
8. Surf, M., Shulman, A.: How safe is it out there? Zeroing in on the vulnerabilities of application security. Imperva Application Defense Center (2004)
9. The elections' atlas of the USA. http://www.uselectionatlas.org/
10. The Parlimental Elections in Estonia (2003), http://www.vvk.ee/r03/yld_kulud.stm
11. Department of Defense Washington Headquarters Services Federal Voting assistance Program. Voting Over the Internet Pilot Project Assessment Report (2001)
12. E-voting System: Overview. Estonian National Electoral Committee (2005) http://www.vvk.ee/elektr/docs/Yldkirjeldus-eng.pdf
13. Triinu Mägi. Practical Security Analysis of E-voting Systems. Msc Thesis. Tallinn University of Technology (2007), http://triinu.net/pro/

Fine-Grained Sticky Provenance Architecture for Office Documents

Takuya Mishina, Sachiko Yoshihama, and Michiharu Kudo[⋆]

IBM Research, Tokyo Research Laboratory
1623-14 Shimo-tsuruma, Yamato, Kanagawa, 242-8502, Japan

Abstract. Current business situations require improved confidentiality and integrity for office documents. However, existing content management systems for office documents lack required security properties such as the $*$-property, or have problems such as label creep. In this paper we propose a meta-data format called *sticky provenance* and a fine-grained information flow control architecture using the sticky provenance. The sticky provenance contains the change history and the labels of an office document in a secure form, and it ensures the verifiability of the change history of the documents in distributed environments. The Provenance Manager, which is a key module of the architecture, reduces the label creep problem of the information flow control models with the sticky provenance. In other words, the sticky provenance and the Provenance Manager can introduce a practical fine-grained information flow control capability to office applications so that we can ensure both the confidentiality and the verifiability of office documents.

1 Introduction

In this paper we tackle the security problems of office documents. To solve the problems, we design a fine-grained information flow control architecture with *sticky provenance*. The sticky provenance is a cryptographically attached *provenance* to office documents. We redefine the term "provenance" as "a protected record of the changes of the content and the security labels of a document," and we define the data structure of the sticky provenance to ensure both the confidentiality and the integrity of the office documents.[1]

There are two main reasons to focus on office document security. One is that the office documents often contain confidential information belonging not only to the owner's organization but also to their customers. The other is that there are gaps between existing content management systems and information flow control models.

Now let us show several concrete examples of threats to office documents.

[⋆] This study was sponsored by the Ministry of Economy, Trade and Industry, Japan (METI) under contract for the New-Generation Information Security R&D Program.

[1] Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product or service names may be trademarks or service marks of others.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 336–351, 2007.
© Springer-Verlag Berlin Heidelberg 2007

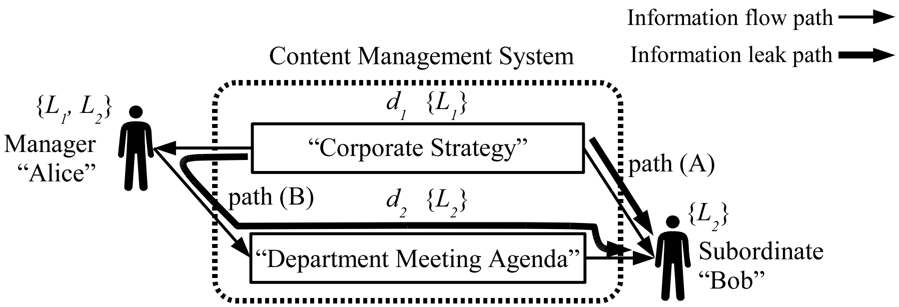


Fig. 1. An example of a content management system and confidential documents

Example 1 (Information leak via legitimate users). Figure 1 shows an example situation of an information leak. The system in Fig. 1 consists of four entities and one system: (1) an office document d_1, which contains company's strategy with label L_1, (2) an office document d_2, which is just an agenda for a group meeting with label L_2, (3) a user called "Alice," who has a clearance for L_1 and L_2 (denoted $\{L_1, L_2\}$), (4) a user called "Bob," who has a clearance for L_2, and (5) a content management system which has a access control capability. If Bob attempts to read d_2 via the path (A) in Fig. 1, the system denies the read access, and therefore d_1 can be secure. However, if Alice copied the data from d_2 to d_1, then Bob can at least read a part of the confidential information belonging to L_1 via the path (B). No malicious user exists in the example, but the documents are threatened with an internal information leak risk.

The Bell-LaPadula model [1], which is one of the Multi-Level Security (MLS) models, is designed for the purpose of controlling information flows to ensure information confidentiality. A system based on the Bell-LaPadula model can prevent the indirect information leaks such as the flow via the path (B) in Fig. 1. However, the Bell-LaPadula model-based systems have some drawbacks and limitations for the purpose of the document management. The following three examples reveal the drawbacks and the limitations.

Example 2 (Information flows cause exclusion of legitimate users). Suppose the system in Fig. 1 has an information flow control capability. If Alice copied data from d_1 to d_2, then the system automatically raises the label of d_2 to $\{L_1, L_2\}$ to comply with the $*$-property of the Bell-LaPadula model. After the copy operation, Bob can no longer read d_2 even if d_2 does not contain information belonging to L_1 (e.g., d_2 is just a harmless summarization of the strategy and it can be disclosed to Bob).

Example 3 (Labeling difficulties). In Example 2, Alice may attempt to remove the label L_1 from d_2. However, in actual cases it is difficult to decide whether or not the labels of the components should be removed because there are plenty of components (paragraphs, shapes, images, etc.) which have various labels in the document.

Example 4 (Security policies cannot be enforced outside of the system). In Example 1, Bob can send some content of d_2 to an external person against the policy on d_2. Even if d_2 is encrypted at the central storage of the content management system, the office application in Bob's client computer will decrypt d_2 before editing it because the office application and the user cannot handle the document in an encrypted form.

Example 5 (The integrity of the history of a document cannot be verified). Suppose an employee of a company, and the employee creates a financial report as an office document. Other employees may update the document several times. Before disclosing the document, the company's CFO (Chief Financial Officer) has to approve it. the CFO can verify the integrity of the latest version of the document with the digital signature of the document, but cannot verify the update history of the document.

The examples reveal that the existing content management systems are not enough to ensure the office document security, and that it is not straightforward to apply the MLS capability to the content management systems. There are security and practicality gaps between the existing content management systems and MLS-capable systems. The practicality gap seems to be a small problem, but actually practicality affects the security of a system seriously. Fine-grained labeling, which is a technique of labeling components instead of labeling its container document for reducing the label creep problem, is one of the solutions to bridge the gaps, but it has several weak points. It requires an implementation in the upper layer of the software stack (e.g., in the applications layer), so we need to implement multiple label management modules for each application. Another weak point is that it introduces a number of costly operations such as label management. In addition to the gaps, both deployment and enforcement of security policies are needed to ensure the office document security in distributed environments.

To solve the problems, we propose a new architecture and a new data structure. Our contributions are listed below:

- **We propose a new paradigm for office document security** (Sect. 3). The paradigm is called *Information flow control with fine-grained sticky provenance*. The key idea of the paradigm is applying the information flow control model with fine-grained labels (Sect. 3.1) to the office applications and the office documents. To define the labels semi-automatically, we employ the notion of *provenance* (Sect. 3.2). The sticky provenance (Sect. 3.3) is an effective meta-data exchange format for the office documents in distributed environments.
- **We design a new architecture for the fine-grained information flow control for office documents** (Sect. 4.1, 4.2, and 4.3). The Provenance Manager is a key module of the architecture, and it serves several roles: label management, linkage of the labels to the components, conversion of data management, and information flow control.
- **We define a data structure for the sticky provenance** (Sect. 4.4) to solve the challenges introduced by the sticky provenance paradigm (Sect. 3.4). The paradigm consists of some ideas: removing sensitive information from the content part of an office document, splitting the meta-data into multiple parts that each of the parts represents a pair of a domain and a confidentiality level. We apply the sticky provenance to the existing data format such as the clipboard data structure and the Open Document Format (ODF).

This paper covers the first step based on our previous work [2], especially in the applications layer provenance monitor and the data exchange format.

2 Problem Statement

2.1 Target Definition and Situation

In this paper we address the security problems of office documents. There are huge numbers of office documents in an organization, and office documents are the main storage form for business information. The office documents have several features: they are updated frequently, exchanged between multiple organizations, and form aggregations of multiple document with sources. The security policies of the documents may be lost in the processes of document exchanges. Such features increase the information leak risk of office documents.

Current business situations such as an increase of business outsourcing, alliances with other companies, and frequent mergers and acquisitions are making the security issues for office documents complicated. Another big concern is compliance. Regulations such as Sarbanes-Oxley Act and Basel II require the integrity of internal documents in an organization.

There are two reasons to restrict the target of our study to the office documents. One is that a fine-grained labeling engine has to be implemented in the applications layer. Therefore, if we implement a fine-grained labeling engine for office applications, we can achieve maximum of efficiency at a minimum of effort. The other is the popularization of open source software and open standard file formats. Both of them improve the interoperability of the fine-grained labeling engine and the sticky provenance. The OpenOffice.org [3] and the Open Document Format (ODF) [4, 5] are the examples of the open source software and the open standard file formats.

2.2 Assumptions and Goal

Before setting the goal of this study, we make the following assumptions:

- The runtime environment for our architecture contain no program that the user does not intend to execute.
- The authorized users may make errors, or the users (except for the trusted subject[2]) may be malicious.
- The information flow channels are limited to files and the clipboard service of operating systems. Other channels such as the shared memory are not assumed as information flow channels.

Now we set the goal and the assuming threats as follows:

- **Ensuring the confidentiality of an office document**. Each document cannot be read except for legitimate users wherever the document is located, and both the content and the meta-data of the document cannot be removed or altered.

[2] The trusted subject is a trustworthy subject who is not restricted by the $*$-property in the Bell-LaPadula model.

- **Ensuring the integrity of an office document**. Each document including its change history can be verifiable. A verifier can verify the integrity of the difference between versions of the document, and the author of each version.

2.3 Problems

There are several problems such as the examples in Sect. 1. Now we state the problems.

Problem 1 (Lack of the $*$-property). Most of the existing content management systems lack the $*$-property, and this may cause confidential information leaks and information contamination. The lack of the $*$-property could allow information to leak from a higher-level document to a lower-level document via the outside of the system boundaries.

Problem 2 (Label Creep). In the Bell-LaPadula model, the label assigned to each information can rise, but cannot fall. Therefore information flows finally cause all of the labels to rise to the least upper bound of the label lattice. This phenomenon is called "label creep," and the label creep lowers the accessibility for legitimate users. In the strict sense, the label creep is necessary to ensure the confidentiality of information. However, indiscriminate label creep may cause unnecessarily label rising (Example 2). The *declassification* [6, 7] (reduce the label for a component in cases where the label is too strict for the component) is required to support the accessibility for the legitimate users.

Problem 3 (Declassification difficulties). Declassification operation can be difficult decisions for trusted subjects as to whether or not an object (document) can be declassified. In the practical situations, the declassification decisions can not be straightforward because there is a lot of information, including paragraphs, shapes, and images.

Problem 4 (Enforcement capabilities). The enforcement capabilities of a content management system are limited to the inside of the system, and it cannot enforce any policies on the documents which have gone outside of the system. The limited enforcement capabilities are irreconcilable with the features of office documents that they are often sent to external organizations.

Problem 5 (The verifiability of the history of a document). According to the legislations, every company has to ensure the integrity of the internal processes. In other words, the company has to ensure the verifiability of the history of internal documents. However, most of the office document formats do not provide the verifiability for the change history.

3 Overview of Our Idea

The core idea of our proposal is *information flow control for office documents based on fine-grained sticky provenance*. This idea is based on an MLS (Multi-Level Security) system which has the $*$-property, and the idea protects the MLS system from the label creep problem and supports declassification operations. Now we show solutions to the problems.

Solution 1 (Fine-grained information flow control for Problem 1, Problem 2, and Problem 3). Assigning a security label to each component (paragraphs, shapes, images, etc.) in each office document, and warn a user if the user is about to produce an information flow causing a label creep problem. The information flow control model reduces the risk of label creep (Example 1), and the rich meta-information given by the fine-grained label supports users in the declassification decisions (Example 2).

Solution 2 (Provenance). Tracking and recording the *provenance* of an office document. This solution supports users to determine the labels of components (Example 3).

Solution 3 (Sticky provenance for Problem 4 and Problem 5). Attaching the provenance data and the security labels to each office document in a cryptographic sense. The content and meta-data are *sealed*[3] so that applications which do not enforce the policy for the document (e.g., existing office application without the extension we implemented) cannot read any data in the document (Example 4). The provenance data is structured to allow users to verify not only the latest revision of the document but also the change history of the document (Example 5).

In the rest of this section we will give an overview of our ideas (e.g., the definition of the term "provenance"), and then we will mention how to realize the solutions in Sect. 4.

3.1 Fine-Grained Labeling

In our architecture, we label the components such as paragraphs, shapes, and images in an office document, whereas most of the existing content management systems label objects at a coarse granularity (such as files). Under the fine-grained labeling, information flow control is performed at the granularity of components, and therefore the office documents are free from the label creep problem. The fine-grained labeling also provides the users with rich meta-information for declassification, and thus the danger of inappropriate declassification can be decreased. The fine-grained labeling engine needs to be implemented as an application extension, not as a middleware extension or as an operating system extension (Sect. 1).

As we mentioned above, the fine-grained labeling capability forces users to provide rich information for the labels. However, it is costly for the users to provide rich meta-information to the labeling engine. To support the users, we introduce a notion of *provenance*. The rich information given by the provenance allows semi-automatic definitions of the labels.

3.2 Provenance

The term *provenance* generally means "the place of origin or earliest known history of something" or "a record of ownership of a work of art or an antique, used as a guide to authenticity or quality."[4] One of the definitions of *provenance* (or *data provenance*)

[3] "Seal" means an encryption operation with keys corresponding to the labels (Sect. 4.4).

[4] Oxford Dictionary of English, 2nd Edition, 2003.

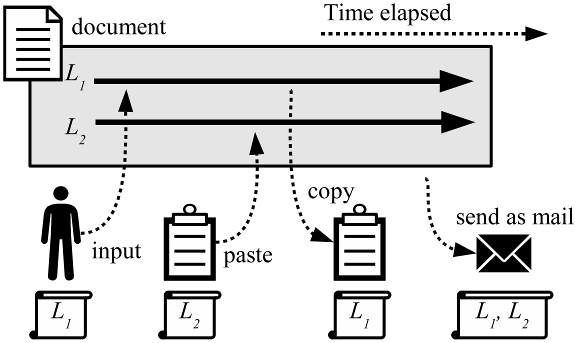


Fig. 2. The notion of the provenance-enabled office document

in the field of computer science is "broadly to refer to a description of the origins of a piece of data and the process by which it arrived in a database" [8]. E-science is the most popular data provenance utilization which aims to make it possible for a third party to verify the results of scientific calculations [9].

In our proposed architecture we redefine *provenance* as "a protected record of the changes of the content and the labels of a document." The reason why we use a different definition is that we need to ensure the confidentiality of office documents in addition to the integrity of them in distributed environments. A typical office document is a snapshot of the latest content, and the history of the information flows has been lost. Some office document formats have the capabilities to record the change history, but meta-data such as the change history of the author information is not preserved.

We extend the office document format into a provenance-enabled format, as shown in Fig. 2. The extended office document has a sequence of records containing author information or label information of the objects related to the information flow as provenance data. The combination of the fine-grained labeling and provenance-enabled data structure supports both confidentiality and declassification. We will show the data structure of the provenance in Sect. 4.4.

3.3 Sticky Provenance Paradigm

The sticky provenance is the provenance that is cryptographically attached to office documents. This notion is derived from the notion of a *sticky policy* [10, 11], suitable for distributed environments. The sticky provenance contains the change history and the labels of an office document in a secure form, and it ensures the verifiability of the change history of document in distributed environments. In contrast to the centralized data management systems, sticky provenance allows the powerful enforceability of policies on the document. For example, if a user opens an office document with the sticky provenance using an office application without the capability for handling the sticky provenance, the user cannot alter or remove any provenance data in the document because the user cannot update the cryptographic link (Sect. 4.4 and Fig. 4).

3.4 Challenges of Fine-Grained Sticky Provenance Paradigm

The sticky provenance paradigm introduces some technical challenges in addition to the problems described in Sect. 2.3.

Challenge 1 (Confidentiality of the provenance data). The provenance data may contain confidential information because it contains a history of content and its labels.

Challenge 2 (Impact for document verifiability). Fine-grained information flow degrades the verifiability of an office document. Only a verifier having a clearance to all of the components in an office document can verify the entire office document.

In the centralized content management systems, meta-data is stored at the central storage, and therefore it can be protected by the access control capability of the system. In contrast to the centralized systems, our architecture need to encrypt the sticky provenance need to protect it against illegal operations. We will define the data structure of the sticky provenance in Sect. 4.4 so that we can solve the challenges.

4 Fine-Grained Sticky Provenance Architecture

Our architecture consists of an office application extension module called *Provenance Manager* and a data structure called *sticky provenance*. In this section we will show them in detail.

4.1 Design Principles

Based on the goal and the assumptions (Sect. 2.2), we set our design principles for the solutions (Sect. 3) as follows:

1. Attaching labels at a fine granularity.
2. Controlling (accepting or denying) the information flows of the components both within an office document and between multiple office documents.
3. Ensuring the enforcement capability of components even if they leave their container office document.

4.2 Architecture Overview

Figure 3 shows the fine-grained sticky provenance architecture. The Provenance Manager, an extension for an office application, plays the key role for the architecture. Provenance Manager serves the following roles:

- Appending security labels to user input (1).
- Managing the visibility and the security label of each office component (2). The Provenance Manager stores the security labels of office components in the Provenance Store (3), and it controls the visibility of office components based on their security label and the document editor's clearance[5]. It also provides graphical user interfaces (4) to show or update the meta-information including the provenance and the security label of each office component.

[5] This can be thought of as information flow control between a document and a document editor.

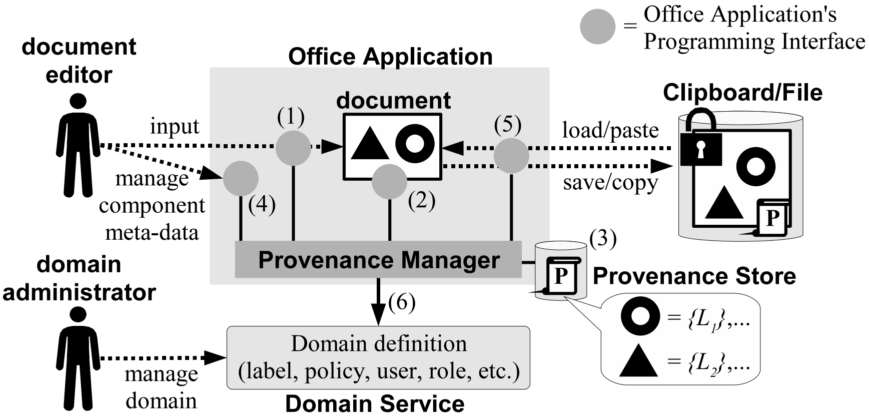


Fig. 3. The fine-grained sticky provenance architecture. The Provenance Manager controls information flows among office documents using the office application's API and external domain services.

- Controlling information flow between an office document and external resources such as the system clipboard and the file system. At the timing of any input or output, the Provenance Manager extracts combines the provenance information from/with an office document (5).
- Use "domain" for the notion of "compartment" in the MLS systems. The Provenance Manager retrieves domain-related information such as label definitions, policies, users, and role definitions from external domain services (6) managed by domain administrators.

4.3 Fine-Grained Labeling

Definition 1. *Define the label L for an office document d ($d = \{c_1, c_2, \ldots, c_n\}$)), and the label L_i for an office component c_i in an office document d as follows.*

$$L_i := \{dom, conf\} \quad (1)$$

$$L := \bigsqcup_{c_i \in d} L_i \quad (2)$$

$\bigsqcup$ denotes the least upper bound of the labels in the document. The definition means that the label of an component can have only one pair of a domain *dom* (e.g., "Sales Department") and a confidentiality level *conf* (e.g., "Confidential") at the same time, and the label of an office document is the least upper bound of the labels of the components it contains. The least upper bound is suitable for the operating-system-layer security label of an office document if the document protection is performed by the operating system which employs the lattice model [12].

4.4 Sticky Provenance

We organize the provenance data as shown in Fig. 4, and Fig. 5 shows an example of the sticky provenance. The data structure definition of the sticky provenance is the key of the proposal to solve the challenges caused by the sticky provenance paradigm. The ideas are as follows:

1. Removing (sanitizing) sensitive information from the content part of an office document. The sensitive information is securely stored in the meta-data part of the office document. We call the meta-data part *sticky provenance*.
2. Splitting the sticky provenance into multiple parts. Each of the parts represents a pair of a domain and a confidentiality level, and it may be "sealed" and signed.

"Seal" means an encryption operation with a domain-specific key which is protected and provided by the domain service in Fig. 3. The seal feature prevents unauthorized users from accessing the confidential office components in a document, and the signatures enable document receivers to verify the integrity of the portions (office components) of the document.

Although the seal operation and the signatures are useful for access control and integrity verification, such cryptographic operations increase performance overhead of the system. To minimize the performance overhead, we organize the components into the sets that each of them contains the components having the same security label, and we perform the cryptographic operations only once per set.

Definition. We define the sticky provenance as follows:

$$\mathit{prov}(\mathit{rev}_\ell) := \left\{\mathit{dom}, \mathrm{Sig}(\mathit{rev}_\ell), \mathrm{Sig}(\mathit{prov}) \;\middle|\; \mathit{dom} \in \mathit{domset}\right\} \tag{3}$$

$$\mathit{dom} := \left\{\mathit{conf}(\mathit{dom}) \;\middle|\; \mathit{conf}(\mathit{dom}) \in \mathit{confset}(\mathit{dom})\right\} \tag{4}$$

$$\mathit{conf}(\mathit{dom}) := \left\{\mathit{change}(\mathit{dom}, \mathit{conf}, t), \mathrm{Sig}\left(\mathit{conf}(\mathit{dom})\right) \;\middle|\; 1 \le t \le \ell\right\} \tag{5}$$

$$\begin{aligned}\mathit{change}(\mathit{dom}, \mathit{conf}, t) := \mathrm{Seal}(\{&\mathit{diff}(\mathit{dom}, \mathit{conf}, t, t-1), \mathit{auth}(\mathit{dom}, \mathit{conf}, t),\\ &\mathit{prev}(\mathit{dom}, \mathit{conf}, t)\}, \mathit{sealpol}(\mathit{dom}, \mathit{conf}))\end{aligned} \tag{6}$$

$\mathit{prov}(\mathit{rev}_\ell)$ denotes the provenance meta-data entry for the ℓ-th revision of the document where *diff* denotes the difference between revisions, *auth* denotes the authorization information, and *prev* denotes the reference to its previous revision.

Seal $(d, \mathit{sealpol}(\mathit{dom}, \mathit{conf}))$ is a seal operation and we define it as follows:

$$\mathrm{Seal}\left(d, \mathit{sealpol}(\mathit{dom}, \mathit{conf})\right) := \begin{cases}\mathrm{Enc}(d, k(\mathit{dom}, \mathit{conf})) & \text{if } \mathit{sealpol}(\mathit{dom}, \mathit{conf}) \text{ is true}\\ d & \text{if } \mathit{sealpol}(\mathit{dom}, \mathit{conf}) \text{ is false}\end{cases} \tag{7}$$

$\mathit{sealpol}(\mathit{dom}, \mathit{conf})$, a part of the domain policy, returns true if the policy requires that the data belonging to $(\mathit{dom}, \mathit{conf})$ have to be sealed where $k(\mathit{dom}, \mathit{conf})$ denotes a symmetric encryption key for the data belonging to $(\mathit{dom}, \mathit{conf})$.

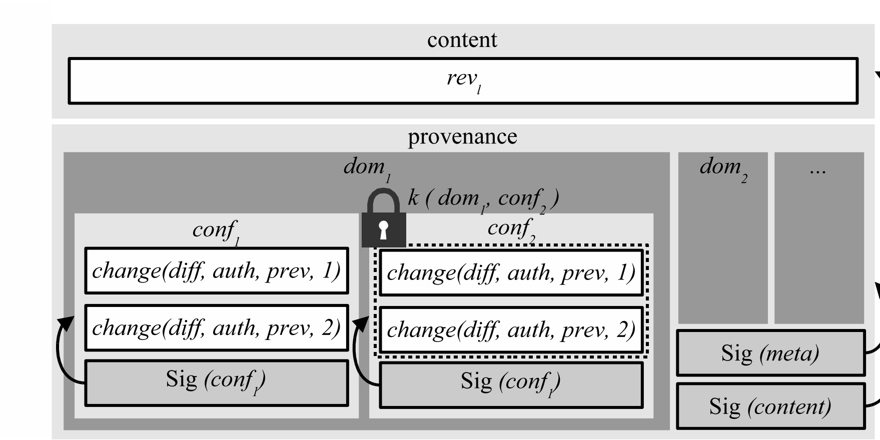


Fig. 4. The data structure of the sticky provenance

```
<prov:Provenance
    xmlns:prov="http://www.trl.ibm.com/provenance/2007/02"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  <prov:DomainProvenance>
    <prov:Domain domainId="dom1" uri="http://www.example.com/domain1/">
      <prov:DomainName>domain1</prov:DomainName>
      <prov:DomainDescription>Domain 1</prov:DomainDescription>
      <prov:DomainPolicy>...</prov:DomainPolicy>
    </prov:Domain>
    <prov:ClassifiedProvenance confidentiality="CONFIDENTIAL">
      <prov:ProvenanceEntry updateDateTime="2007-03-01T20:46:07.921+09:00">
        <prov:AuthInfo>
          <prov:UserName>tmishina</prov:UserName>
          <prov:RoleName>member</prov:RoleName>
          <prov:Authority type="directory" uri="ldap://example.com/" />
        </prov:AuthInfo>
        <prov:Difference>
          <prov:Component componentId="18b9f214-0467-4542-ac7e-035b016d0934">
            ...
          </prov:Component>
        </prov:Difference>
      </prov:ProvenanceEntry>
      <ds:Signature>...</ds:Signature>
    </prov:ClassifiedProvenance>
  </prov:DomainProvenance>
  <ds:Signature>...</ds:Signature>
</prov:Provenance>
```

Fig. 5. An example of the sticky provenance in XML. `prov:DomainProvenance`, `prov:ClassifiedProvenance`, and `prov:ProvenanceEntry` represent *dom*, *conf*(*dom*), and *change*(*dom*, *conf*, *t*) respectively. If the policy (`prov:DomainPolicy`) requires a seal operation for the confidentiality `CONFIDENTIAL`, the contents of `prov:ProvenanceEntry` will be sealed.

Applying to Document Fragment. We call the data format for in-memory data exchange *Document Fragment*, especially for the data exchange through the clipboard service provided by operating systems. In a Document Fragment format, the data to be "copied" is replaced by its provenance meta-data to ensure the enforcement capability on the outside of the office application.

The OpenOffice.org converts the target components into its native XML (eXtensible Markup Language) format at the timing of a copy operation in addition to the common

```
<office:document xmlns:office="http://openoffice.org/2000/office"
 xmlns:prov="http://www.trl.ibm.com/provenance/2007/02">
 <office:meta>
   <prov:Provenance>...</prov:Provenance>
 </office:meta>
 <office:automatic-styles>
   <style:style style:name="gr1">
     <style:properties prov:componentId="18b9f214-0467-4542-ac7e-035b016d0934" />
   </style:style>
 </office:automatic-styles>
 <office:body>
   <draw:page>
     <draw:rect draw:style-name="gr1"/>
   </draw:page>
 </office:body>
</office:document>
```

Fig. 6. An example of the ODF file containing the sticky provenance data as a `prov:Provenance` element. Each office component is an empty (sanitized) element, and it has a user-defined attribute `prov:componentId` to be identified by the Provenance Manager.

data types such as Windows Bitmap (`.bmp`). We use the XML format as the source content of the Document Fragment to generate the custom data.

Applying to the Open Document Format. The Open Document Format (ODF) is a standardized format for office documents, and it is the native format for the OpenOffice.org. An ODF file is a ZIP archive which consists of several XML files and binary files including the content part (`content.xml`), the meta-data part (`meta.xml`), and the other files such as inserted images. Each of the components such as paragraphs and shapes is represented as a standardized XML element.

We define a new format called *Open Document Format with Sticky Provenance* (ODFP) as a combination of a sanitized original ODF content and its provenance. The meta-data ($prov(rev_\ell)$ in (3)) is attached as a child element of the `meta` element defined in the ODF standard as shown in Fig. 6.

4.5 Information Flow Control Rules

We assign three choices to the Provenance Manager for an inbound information flow such as a paste operation or a file load operation as follows.

accept. If a user has a clearance for the object of the information flow, and if the target office document has a higher label than that of the object.

warn (conditional accept). If a user has a clearance for the object of the information flow but there exists a component which has a label different from that of the object in the target office document, the Provenance Manager warns the user of label creep. If the user accept the warning, the label of the office document automatically rises.

deny. If a user does not have a clearance for the target object, the information flow is automatically denied because the user fails to unseal the target object.

Outbound information flows such as copy operations and file storing operations are always accepted because all of the information is sealed (Sect. 4.4). Security decisions and their enforcement will occur at the time of inbound information flow operations.

5 Related Work

Content Management Systems such as an Enterprise Content Management (ECM) systems or a Digital Rights Management (DRM) systems manage documents in a central repository and controls access to the documents. The products for ECM and DRM are suitable for documents that are managed only in one organization[6], or documents that are updated infrequently. From the viewpoint of our problem statement, a Content Management System is not suitable because it does not have the $*$-property, and therefore it cannot prevent users from doing unintended operations that cause information leaks. A *Collaborative environment* is our target environment. Tolone et al. [13] clarified technical problems with access control for such environments. Bertino et al. [14] proposed an infrastructure to ensure the security for XML documents which are exchanged among different parties using XPath [15]. Their proposed infrastructure and its data structures also ensure both the confidentiality and the integrity of documents. In contrast to their proposal, our architecture employs label-based information flow control instead of the (X)path-based access control. Pan et al. [16] proposed middleware which enables access control to data that is stored in a centralized database and shared across different organizations. Jin et al. [17] also proposed a framework to enable secure sharing of scientific data over a distributed environment. *Provenance* is mainly discussed in the area of e-science as mentioned in Sect. 3.2. There are general discussions [9, 18, 19] about the verifiability of e-science results, and some concrete studies [20]. Recently, update operations [21, 22] and workflow integration [23] are hot topics.

Information flow control frameworks for the operating systems such as SELinux [24] can enforce security policies around a computer platform. However, policy granularity of the OS layer framework is more coarse than that of an application layer information flow framework. *Language-based information flow control* [25, 26, 27] enables considerably fine-grained information flow control, and such approaches can be integrated with our work. *XML Access Control* is one of the most active research areas, and there is a lot of previous work [28, 29] including XACML [30], an open standard to express access control policies. Mazzoleni et al. [31] proposed an algorithm to integrate multiple XACML policies to enable collaborative works between multiple parties. Combining access control and *XML Versioning* [32, 33] is another active research area. Iwaihara et al. [34] proposed a new expression called "XVerPath" to designate a version-aware XML element. Fine-grained access control is also important for XML documents [35].

Our architecture requires partial verifiability because the content of the document fragment is an aggregate consisting of the encrypted data belonging to each domain. *Sanitizable signature algorithms* [36, 37, 38] allows documents to provide a capability for partial verifiability. The sticky provenance provides the verifiability for the documents even if the document consists of encrypted fragments.

Reasonable performance is always demanded of practical security systems. Effective diff creation [39], access control performance improvement [40], and effective change detection [41] can be integrated with our work to improve its performance.

[6] Some products provide document administrators to enforce security policies for the documents which have gone out of the organization's computers.

6 Conclusion

In this paper we proposed a new paradigm *information flow control for office documents based on fine-grained sticky provenance* that improves the confidentiality and the integrity for office documents while reducing the label creep problem and the declassification difficulties. We also design a fine-grained information flow control architecture based on the paradigm, and define the data structure of the sticky provenance.

Currently we are working to implement the Provenance Manager as an extension for OpenOffice.org. When the implementation is finished, we need to evaluate the performance overhead of the implementation. We also need further studies on the policy definition (e.g., *domset*, *confset*, *sealpol*(*dom*, *conf*), and declassification policies) and key management. The notion of Trusted Virtual Domains [42] can also be integrated with our architecture to support the domain management and the software component validation. The other important future project will be a collaboration with reference monitors on the operating system layer to retrieve the meta-information originated in the operating system layer.

References

[1] Bell, D.E., LaPadula, L.J.: Secure computer system: Unified exposition and multics interpretation. Technical Report MTR-2997 Rev. 1, MITRE Corporation, (March 1976)

[2] Mishina, T., Watanabe, Y., Katsuno, Y., Yoshihama, S.: Semantic fine-grained data provenance tracking (works in progress). In: Proc. of 22nd Annual Computer Security Applications Conference (2006)

[3] Openoffice.org. http://www.openoffice.org/

[4] OASIS open document format for office applications (OpenDocument) TC. http://www.oasis-open.org/committees/office/

[5] Open document format for office applications (OpenDocument) v1.0, 2006. ISO/IEC 26300:2006.

[6] Ferrari, E., Samarati, P., Bertino, E., Jajodia, S.: Providing flexibility in information flow control for object oriented systems. In: Proc. of the 1997 IEEE Symposium on Security and Privacy, pp. 130–140 (1997)

[7] Chong, S., Myers, A.C.: Security policies for downgrading. In: Proc. of the 11th ACM conference on Computer and communications security, pp. 198–209 (2004)

[8] Buneman, P., Khanna, S., Tan, W.C.: Data provenance: Some basic issues. In: Kapoor, S., Prasad, S. (eds.) FST TCS 2000. LNCS, vol. 1974, pp. 87–93. Springer, Heidelberg (2000)

[9] Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Record 34, 31–36 (2005)

[10] Karjoth, G., Schunter, M., Waidner, M.: The platform for enterprise privacy practices—privacy enabled management of customer data. In: The 2nd Workshop on Privacy Enhancing Technologies (2002)

[11] Mont, M., Pearson, S., Bramhall, P.: Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In: Proc. of the 14th International Workshop on Database and Expert Systems Applications (2003)

[12] Denning, D.E.: A lattice model of secure information flow. Communications of the ACM 19(5), 236–243 (1976)

[13] Tolone, W., Ahn, G.J., Pai, T., Hong, S.P.: Access control in collaborative systems. ACM Computing Surveys 37(1), 29–41 (2005)

[14] Bertino, E., Mella, G., Correndo, G., Ferrari, E.: An infrastructure for managing secure update operations on XML data. In: Proc. of the eighth ACM symposium on Access control models and technologies, pp. 110–122 (2003)

[15] XML path language (xpath) version 1.0, W3C recommendation 16 november (1999), http://www.w3.org/TR/xpath

[16] Pan, C.C., Mitra, P., Liu, P.: Semantic access control for information interoperation. In: Proc. of the eleventh ACM symposium on Access control models and technologies, pp. 237–246 (2006)

[17] Jin, J., Ahn, G.J.: Role-based access management for ad-hoc collaborative sharing. In: Proc. of the eleventh ACM symposium on Access control models and technologies, pp. 200–209 (2006)

[18] Tan, W.C.: Research problems in data provenance. IEEE Data Engineering Bulletin 27(4), 45–52 (2004)

[19] Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, Springer, Heidelberg (2000)

[20] The EU provenance project. http://twiki.gridprovenance.org/bin/view/Provenance/

[21] Buneman, P., Chapman, A., Cheney, J.: Provenance management in curated databases. In: Proc. of the 2006 ACM SIGMOD international conference on Management of data, pp. 539–550 (2006)

[22] Buneman, P., Chapman, A., Cheney, J., Vansummerenn, S.: A provenance model for manually curated data. In: Proc. of the International Provenance and Annotation Workshop (2006)

[23] Tan, V., Groth, P., Miles, S., Jiang, S., Munroe, S., Tsasakou, S., Moreau, L.: Security issues in a SOA-based provenance system. In: Proc. of the International Provenance and Annotation Workshop (2006)

[24] Loscocco, P., Smalley, S.: Integrating flexible support for security policies into the linux operating system. In: Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference (2001)

[25] Myers, A.C.: JFlow: Practical mostly-static information flow control. In: Symposium on Principles of Programming Languages, pp. 228–241 (1999)

[26] Sabelfeld, A., Myers, A.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications 21(1), 5–19 (2003)

[27] Yoshihama, S., Kudoh, M., Oyanagi, K.: Language-based information flow control in dynamic approach. IBM Research Report TR0694, IBM Research (2006)

[28] Bertino, E., Castano, S., Ferrari, E., Mesiti, M.: Controlled access and dissemination of XML documents. In: Proc. of the 2nd international workshop on Web information and data management, pp. 22–27 (1999)

[29] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: Securing XML documents. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, Springer, Heidelberg (2000)

[30] OASIS eXtensible Access Control Markup Language (XACML) TC. http://www.oasis-open.org/committees/xacml/

[31] Mazzoleni, P., Bertino, E., Crispo, B., Sivasubramanian, S.: XACML policy integration algorithms. In: Proc. of the eleventh ACM symposium on Access control models and technologies, pp. 219–227 (2006)

[32] Chatvichienchai, S., Anutariya, C., Iwaihara, M., Wuwongse, V., Kambayashi, Y.: Towards integration of XML document access and version control. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, Springer, Heidelberg (2004)

[33] Chatvichienchai, S., Iwaihara, M.: Detecting information leakage in updating XML documents of fine-grained access control. In: Proc. of Database and Expert Systems Applications, pp. 286–296 (2006)

[34] Iwaihara, M., Chatvichienchai, S., Anutariya, C., Wuwongse, V.: Relevancy based access control of versioned XML documents. In: Proc. of the tenth ACM symposium on Access control models and technologies, pp. 85–94 (2005)

[35] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for XML documents. Transactions on Information and System Security 5(2), 169–202 (2002)

[36] Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signature. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, Springer, Heidelberg (2005)

[37] Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E88-A(1), 239–246 (2005)

[38] Miyazaki, K., Hanaoka, G., Imai, H.: Digitally signed document sanitizing scheme based on bilinear maps. In: Proc. of the 2006 ACM Symposium on Information, computer and communications security, pp. 343–354 (2006)

[39] Rönnau, S., Scheffczyk, J., Borghoff, U.M.: Towards XML version control of office documents. In: Proc. of the 2005 ACM symposium on Document engineering, pp. 10–19 (2005)

[40] Carminati, B., Ferrari, E.: AC-XML documents: improving the performance of a web access control module. In: Proc. of the tenth ACM symposium on Access control models and technologies, pp. 67–76 (2005)

[41] Wang, Y., DeWitt, D.J., Ca, J.Y.: X-Diff: An effective change detection algorithm for XML documents. In: Proc. of the 19th International Conference on Data Engineering, pp. 519–530 (2003)

[42] Katsuno, Y., Kudo, M., Watanabe, Y., Yoshihama, S., Perez, R., Sailer, R., van Doorn, L.: Towards multi layer trusted virtual domains. In: Proc. of The Second Workshop on Advances in Trusted Computing (2006)

Secure Anonymous Communications with Practical Anonymity Revocation Scheme

Koji Chida, Osamu Shionoiri*, and Atsushi Kanai

NTT Corporation, Tokyo, Japan
chida.koji@lab.ntt.co.jp

Abstract. We propose an anonymous communication scheme using a multistage cryptographic anonymous proxy to make malicious users traceable. For the purpose of preventing illegal use of an anonymous channel, the proposed scheme can identify an illegal anonymous user or his accomplice who joins a conspiracy if a threshold number of third parties or more agrees on anonymity revocation. Expected applications of the proposed scheme include real-time web services over the Internet in which both user anonymity and preventing illegal use are desired such as anonymous bulletin board services, auctions, and peer-to-peer file exchange. Moreover, we implemented the proposed scheme to verify the degree of practicality. The implementation results include that the case for the received data of 100 KB from an HTTP request and a ten-stage relay for the anonymous proxy on an Intel Pentium 4 3.0 GHz PC took approximately 1 sec to receive, and the received data of 1 MB and a two-stage relay required 2 sec.

1 Introduction

1.1 Background

In the TCP/IP based World Wide Web, there is apprehension that on the basis of IP addresses, malicious web site administrators or eavesdroppers will invade the privacy of users or launch attacks on user computers. An anonymous proxy, which accesses web sites instead of the users and conceals the IP address of each user terminal from the sites, is an effective tool to prevent these types of security violations. Furthermore, if a multistage cryptographic anonymous proxy (or router) such as Chaum's Mix-net[4] or Onion Routing[9] is employed for anonymous communications, even for each proxy administrator the information pertaining to "who sent what" or "who accessed where" can be concealed and therefore the user security level can be relatively increased.

However, anonymous communications using such proxies may foster user fraud. If a relay proxy is corrupt or administered abroad, it may be difficult to investigate thoroughly the specifics of an illegal user. Moreover, since many existing anonymous proxies rely on volunteers, mismanagement by administrators should be carefully considered.

[*] Presently, the author is with NTT West Corporation.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 352–364, 2007.
© Springer-Verlag Berlin Heidelberg 2007

From the above observation, it is desirable that anonymous communication schemes provide some mechanism for restraining fraudulent user behavior. A practical and technical solution to the problem would be to provide an anonymity revocation mechanism to anonymous communications. That is, a mechanism in which only a system manager (or distributed system managers) can identify an anonymous user.

1.2 Related Work

There are two approaches to provide revocable anonymity to a communication scheme using a multistage cryptographic anonymous proxy.

One is to backtrack the route with the help of all the relay proxies. In such a case, all proxies retain a communication log and in particular the proxy connected to a user terminal must identify the user. However, one problem with this approach is the reliability of the relay proxies or their administrators as stated in Sec. 1.1.

The other approach is to employ a group signature scheme. The group signature scheme, which was introduced by Chaum and Heyst[5], allows a group member to sign messages anonymously on behalf of the group. Most group signature schemes also provide both anonymity and membership revocation mechanisms. However, existing group signature schemes with practical revocation mechanisms [2,1,3,10,8] impose heavy signing and verification costs. To the best of our knowledge, [8] is the most efficient one with regard to the signing and verification costs. Specifically, the scheme requires 36 and 21 multi-exponentiations in $\mathbb{Z}/N\mathbb{Z}$ for signing and verification, respectively, where N is an RSA composite number. When a group signature scheme is employed in a communication scheme using a multistage anonymous proxy, a user terminal generates a group signature for messages and the receiver terminal verifies the signature. A potential problem with this approach is the difficulty of preventing DoS attacks. If a corrupted user sends a large amount of messages without the group signature to a receiver using an anonymous channel, an investigating authority becomes increasingly harder to identify the user compared to the case that a normal channel was used. To prevent this kind of attacks, the first relay proxy additionally has to authenticate a user by an authentication protocol using a group signature and preserve the authentication log.

1.3 Our Contribution

We propose a new approach to provide a revocable anonymity mechanism to the communication scheme using a multistage cryptographic anonymous proxy. The proposed scheme employs not a group signature scheme but a mixture of a general signature scheme and threshold cryptosystem to revoke anonymity. Unlike the group signature based schemes, in the proposed scheme only a receiver preserves a certain part of communication data to identify the anonymous user or his accomplice who improperly supports the user communication. A threshold cryptosystem[7] consists of an encryption function and the corresponding decryption function in which a secret key is distributed to multiple parties, and if a threshold number of parties who retain the distributed keys cooperate, decryption is possible.

Due to the mixture model of the proposed scheme, any user or his accomplice can be specified if a threshold number, t, of k third parties cooperate, and conversely as long as t third parties do not conspire, the proposed scheme provides anonymity for every user equivalent of the scheme that employs a multistage cryptographic anonymous proxy.

Moreover, we report on the implementation results of the proposed scheme to verify the degree of practicality. The results verify that generally the performance is sufficient for practical purposes. For example, the case for the received data of 100 KB from an HTTP request and a ten-stage relay for the anonymous proxy on an Intel Pentium 4 3.0 GHz PC took approximately 1 sec to receive, and the received data of 1 MB and a two-stage relay required 2 sec. Thus, the proposed scheme is considered available for real-time web services in which both user anonymity and preventing illegal use are desired, for example, anonymous bulletin board services, auctions, and peer-to-peer file exchange.

2 Underlying Scheme

There are three types of anonymous communications using a multistage cryptographic anonymous proxy. One is an *asynchronous* scheme typified by Onion Routing. The asynchronous scheme targets real-time anonymous communications by complexifying the communication route going through some relay proxies, but it is assumed that monitoring of a certain network is intractable. Second is a *synchronous* scheme typified by Chaum's Mix-net. The synchronous scheme aims to establish anonymous communications against a strong adversary who can observe the entire network by shuffling and re-encrypting multiple messages. The scheme provides anonymity among a certain group in the presence of the adversary under general cryptographic assumptions. However, it is not suitable for real-time communication applications because of shuffling multiple messages. The last is a *semi-asynchronous* scheme that is a hybrid mode consisting of asynchronous and synchronous schemes for providing both strong anonymity and fast anonymous communications. As stated in [6], the semi-asynchronous scheme operates such that every relay proxy outputs either two messages with shuffling by storing the previous input ciphertext or only the stored message by being a certain time. Please see [6] for details. On the other hand, all of the schemes depends on the reliability of the relay proxies, that is, the anonymity is lost immediately if all relay proxies conspire.

The proposed scheme employs a route-hiding technique based on Onion Routing for real-time anonymous communications. We first give an example of Onion Routing below.

Indicated below is a procedure for a message, msg, sent from a user terminal to a receiver terminal, R, via two relay proxies, M_1 and M_2. Here, Addr_1, Addr_2, and Addr_3 are the published addresses for M_1, M_2, and R, respectively, and $\mathcal{E}_i$ represents the encryption function of proxy M_i.

1. A user terminal performs the following.
   (a) Generate multiplexed ciphertext $C_0 \overset{\text{def}}{=} \mathcal{E}_1(\mathsf{Addr}_2 \| \mathcal{E}_2(\mathsf{Addr}_3 \| msg))$ (here, "$\|$" represents data concatenation).
   (b) Send C_0 to M_1 according to Addr_1.
2. The first proxy, M_1, performs the following.
   (a) Decrypt C_0 and obtain $(\mathsf{Addr}_2 \| \mathcal{E}_2(\mathsf{Addr}_3 \| msg))$.
   (b) Send a portion of the decrypted result, $C_1 \overset{\text{def}}{=} \mathcal{E}_2(\mathsf{Addr}_3 \| msg)$, to M_2 according to Addr_2, which is also a portion of the decryption result.
3. The second proxy, M_2, performs the following.
   (a) Decrypt C_1 and obtain $(\mathsf{Addr}_3 \| msg)$.
   (b) Send a portion of the decrypted result, msg, to R according to Addr_3, which is also a portion of the decryption result.

In the above protocol, a user can select relay proxies and their order for message delivery. This aspect of routing flexibility is superior in terms of the privacy of the route even for proxy administrators.

If $\mathcal{E}_i$ represents a public key encryption function, then the decryption procedure becomes obvious. However, if we consider a large message, this method is not efficient. A hybrid encryption protocol is desirable in which a public key encryption is performed to obtain session keys and a message is encrypted with the session keys using a symmetric key encryption algorithm. In that case, the previously described multiplexed ciphertext C_0 can be the form, for example, as $C_0 \overset{\text{def}}{=} \mathcal{E}_1(K_1) \| E_{K_1}(\mathsf{Addr}_2 \| \mathcal{E}_2(K_2) \| E_{K_2}(\ \mathsf{Addr}_3 \| msg))$ where E_K is a symmetric key encryption function using session key K, and K_i is a session key between a user terminal and M_i.

3 Proposed Scheme

3.1 Model

The proposed scheme basically supports semi-asynchronous and asynchronous modes for real-time anonymous communications due to the Onion Routing protocol as stated in the previous section. The entities in the proposed scheme are user U, which hopes to protect the privacy of the user; anonymous proxy group $\mathsf{M} = \{\mathsf{M}_1, \ldots, \mathsf{M}_m\}$, which provides user anonymity; receiver R, which receives the message from the anonymous user; and third party group $\mathsf{T} = \{\mathsf{T}_1, \ldots, \mathsf{T}_k\}$, which performs anonymity revocation when fraud of an anonymous user is detected. We suppose that many U and R exist. Note here that only in the case that $t(\leq k)$ parties cooperate from among T, the user anonymity can be invalidated. We denote the collaborative subset consists of t parties as T' hereafter.

The proposed scheme is divided into two processes, the anonymous communication and the anonymity revocation as shown in Figs. 1 and 2, respectively.

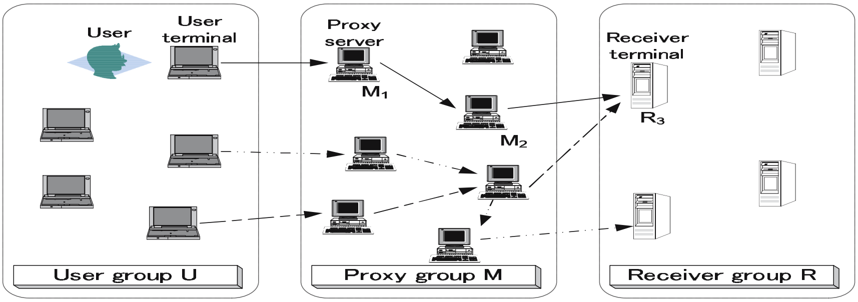


Fig. 1. An overall picture for the proposed anonymous communication scheme

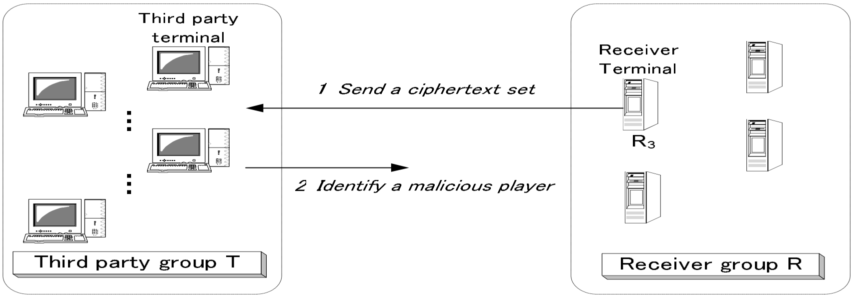


Fig. 2. An overall picture for the proposed anonymity revocation scheme

3.2 Anonymous Communication

We first present the processing for anonymous communications. Here, $\mathcal{E}_i$ and $\mathcal{P}$ are threshold encryption functions such that M_i can decrypt ciphertext $\mathcal{E}_i(\cdot)$ and T' can decrypt ciphertexts $\mathcal{E}_i(\cdot)$ and $\mathcal{P}(\cdot)$. As practical functions for $\mathcal{E}_i$ and $\mathcal{P}$, we implemented hybrid encryption functions based on [6]. The implementation results are given in Sec. 5. Also, U and M_i in advance maintain signature generation functions $\mathcal{S}_0$ and $\mathcal{S}_i$, respectively, and their corresponding verification functions are $\mathcal{V}_0$ and $\mathcal{V}_i$, respectively. We suppose that every signature is publicly linked with a signer, for example, using a certificate.

The processing is as follows.

1. U performs the following.
   (a) Select relay proxies and their order for message delivery.
   (b) Acquire the address of R, Addr_{n+1} where n is the number of relay proxies, and that of M_i, Addr_i $(i = 1, \ldots, n)$.
   (c) For message $C_n \stackrel{\text{def}}{=} msg$, compute $C_i \stackrel{\text{def}}{=} \mathcal{E}_{i+1}(\mathsf{Addr}_{i+2} \| C_{i+1})$ for $i = n-1, \ldots, 0$.
   (d) Compute $\sigma_0 \stackrel{\text{def}}{=} \mathcal{S}_0(C_0)$ and send (C_0, σ_0) to M_1 according to Addr_1.

2. For $i = 1, \ldots, n$, M_i performs the following.
   (a) Verify that $\mathcal{V}_{i-1}(C_{i-1} \| \Gamma_{i-1}, \sigma_{i-1}) = 1$ holds[1] (Γ_0: null).
   (b) Decrypt C_{i-1} and obtain $(\mathsf{Addr}_{i+1} \| C_i)$.
   (c) Compute $\Gamma_i \overset{\text{def}}{=} \mathcal{P}(C_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1})$ and $\sigma_i \overset{\text{def}}{=} \mathcal{S}_i(C_i \| \Gamma_i)$, and send $(C_i, \Gamma_i, \sigma_i)$ to M_{i+1} (if $i = n$, then R) according to Addr_{i+1}.
3. R performs the following.
   (a) Verify that $\mathcal{V}_n(C_n \| \Gamma_n, \sigma_n) = 1$ holds.
   (b) Save input $(C_n, \Gamma_n, \sigma_n)$ and execute processing on message $C_n (= msg)$.

3.3 Anonymity Revocation

Next, we describe the processing for anonymity revocation. The process is as follows.

1. R transmits $(C_n, \Gamma_n, \sigma_n)$ to T, and requests disclosure of the originator who sent message C_n $(= msg)$.
2. T' $(\subseteq \mathsf{T})$ decides on the need for user disclosure based on some reason or another, and if disclosure is determined, T' performs the following.
   (a) Verify that $\mathcal{V}_n(C_n \| \Gamma_n, \sigma_n) = 1$ holds. If not, the request from R is dismissed and the process is terminated.
   (b) For $i = n, \ldots, 1$, the following is performed.
      i. Decrypt $\Gamma_i (= \mathcal{P}(C_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1}))$ and obtain the result, $(C_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1})$.
      ii. Decrypt $C_{i-1} (= \mathcal{E}_{i-1}(\mathsf{Addr}_{i+1} \| C_i))$.
      iii. Verify that $\mathcal{V}_{i-1}(C_{i-1} \| \Gamma_{i-1}, \sigma_{i-1}) = 1$ and the decryption result of C_{i-1} equals $(\mathsf{Addr}_{i+1} \| C_i)$ holds[2]. If not, M_i is determined as the criminal (i.e., an accomplice of the anonymous user) and the process is terminated.
3. The illegal user is specified from (C_0, σ_0).

3.4 Efficiency

In the present anonymity revocation scheme, generally the communication channel is retraced using the receiver as the starting point to the illegal user. In the proposed scheme however, the relay proxies are not required to take part in the anonymity revocation process due to a set that consists of messages, ciphertexts, and signatures, $(C_n, \Gamma_n, \sigma_n)$. This characteristic is very advantageous if anonymous proxies are constructed by peer nodes such as in peer-to-peer networks. Also, as described before, all entities except R do not have to preserve any log data while in a group signature based scheme a first relay proxy has to preserve the authentication log with a sender.

Additional operations by U and R compared to the operation in the underlying scheme described in Sec. 2 are just one-time signing and verification, respectively. This is much more efficient compared to a group signature based scheme

[1] Here, verification function $\mathcal{V}(C, \sigma)$ will return 1 only when (C, σ) satisfies $\sigma = \mathcal{S}_i(C)$.

[2] Addr_{i+1} and C_i are already restored to T' before decrypting C_{i-1}.

described in Sec. 1.2. On the other hand, M_i also generates Γ_i as an additional operation, however, the operation is considered to be reasonable if the encryption function of Γ_i, $\mathcal{P}$, is a hybrid encryption scheme, as described in Sec. 5.

4 Security Analysis

4.1 Anonymity

In order to discuss the anonymity of the proposed anonymous communication processing, we first define a security requirement regarding anonymity as follows.

Definition 1 (Anonymity). *Consider an anonymous communication model with an anonymity revocation mechanism consisting of user* U*, which hopes to protect the privacy of the user, anonymous proxy group* $\mathsf{M} = \{\mathsf{M}_1, \ldots, \mathsf{M}_m\}$*, which provides user anonymity, receiver* R*, which receives the message from the anonymous user, third party group* $\mathsf{T} = \{\mathsf{T}_1, \ldots, \mathsf{T}_k\}$*, which performs anonymity revocation when fraud of an anonymous user is detected, and communication channel* CC *connected between any two players above. Suppose adversary* $\mathcal{A}$ *who tries to find out the linkage between a user and a message sent to* R *through* CC *but can corrupt neither at least one proxy,* M_j*, nor equal or more than a threshold number of third parties and cannot monitor a path in* CC *between from* M_j *to* M_{j+1} *(or* R *if* $j = n$*). Then, we call the communication channel anonymous if* $\mathcal{A}$ *cannot find out the linkage.*

Suppose that the encryption functions used in the proposed scheme, $\mathcal{E}_i$ and $\mathcal{P}$, are semantically secure. Then, we see that all data Γ_i and C_i for $i \neq j$ provide no information in regard to Addr_{j+1}, which is the address of M_{j+1} or R, due to the encryption, and of course every signature, σ_i, for $i \geq 1$ provides no linkage. In addition, $\mathcal{A}$ cannot find out the linkage between C_u and C_v for any $u(< j)$ and $v(> j)$ because the linkage between C_{j-1} and C_j, which are intermediates between C_u and C_v, is lost for $\mathcal{A}$ due to the assumptions of $\mathcal{E}_i$ and that $\mathcal{A}$ cannot corrupt M_j. Similarly, we see that the linkage between Γ_u and Γ_v is intractable for $\mathcal{A}$. This means the linkage between the data sent from a user terminal, (C_0, σ_0), and the data sent to a receiver terminal, $(C_n, \Gamma_n, \sigma_n)$, is broken for $\mathcal{A}$. Therefore, $\mathcal{A}$ cannot find out the linkage between a user and a message sent to R and we reach the following theorem regarding Definition 1.

Theorem 1. *The communication channel constructed by the proposed scheme is anonymous if the encryption functions,* $\mathcal{E}_i$ *and* $\mathcal{P}$*, are semantically secure.*

4.2 Robustness of Anonymity Revocation

Next, we discuss the proposed anonymity revocation processing. It is clear that the user identifier is properly output in the processing if the user, proxies, receiver, and third parties perform the operation correctly. That is to say that integrity is satisfied. Our concern here is robustness of the proposed anonymity revocation processing. Specifically, we address the following security requirement.

Definition 2 (Robustness of Anonymity Revocation). *Consider an anonymous communication model with an anonymity revocation mechanism consisting of* $\mathsf{U}, \mathsf{M}, \mathsf{R}, \mathsf{T}$*, and* CC *similar to them in Definition 1. Then we call the revocation mechanism of a scheme based on the model robust if a corrupted player involved with an illegal communication can be identified with overwhelming probability.*

Definition 2 above is, in other words, the revocation mechanism is required to identify a corrupted player except negligible error based on cryptographic operations.

The following theorem is obtained regarding Definition 2.

Theorem 2. *The anonymity revocation mechanism of the proposed scheme is robust if,*

(i) *equal or more than a threshold number of third parties* T' *perform his/her operation correctly,*
(ii) *the threshold cryptosystem,* $\mathcal{P}$*, is non-malleable,*
(iii) *correctness of decrypting operation of* $\mathcal{P}$ *by a third party is verifiable to all the other parties,*
(iv) *the signature function,* $\mathcal{S}_i$*, is unforgeable,*
(v) *and an adversary cannot corrupt at least one relay proxy.*

Proof. To prove the theorem, we consider possible illegal acts by user, proxies, receiver, and third parties. We can see easily that an incorrect act by a third party and receiver can be prevented by the assumptions (iii) and (iv), respectively. Therefore, in regard to third party and receiver impropriety, the proposed anonymity revocation processing is said to be robust. Next, we consider the fraud by a user or proxy. Neither a user nor proxy can repudiate the sending data because of their signature, however, a proxy may disguise the received data. The received data of proxy M_i are encrypted as Γ_i and the data are correctly decrypted with the cooperation of T' by the assumption (i). Also, C_i, which is a portion of the data sent by M_i, must be a portion of the decrypted result of C_{i-1}, which are data that M_i claims as a portion of the received data, due to the assumption (iii). This means that adversary M_i cannot falsify the received data unless M_i can obtain $C'_{i-1}(\neq C_{i-1})$ from an uncorrupted relay proxy such that a portion of the decryption result of C'_{i-1} is C_i. We see that the success probability of this fraud by M_i is negligible due to the assumptions (ii) and (v). Therefore, the identifier of the illegal user, σ_0, is obtained or the interference by a corrupted proxy is detected by performing the proposed anonymity revocation protocol based on the assumption (i). □

The assumptions of Theorem 2 are considered realistic, however, third parties may identify not an illegal user but a proxy manager who improperly supports the user communication. This property of the scheme is somewhat weaker than a group signature based scheme, which can identify an illegal user as described in Sec. 1.2.

5 Implementation

5.1 Overview

The implemented system establishes portal site P in addition to other entities described in Sec. 3.1, which discloses the addresses and public keys of M_i and R and keeps a user revocation list to manage membership. P can easily and dynamically control the access of each user based on the user revocation list as opposed to the group signature based scheme.

A user first inputs the address for P, and after password authentication, the user can start to use anonymous web services. Due to the password authentication, the user can access services from various terminals by merely entering a password. Note that, P cannot know where the user tries to communicate due to a multistage anonymous proxy.

In the reply process from R to U, the reply route follows the reverse of the transmission route. Each M_i reuses the generated session key at the time of transmission and only needs to encrypt the received data from M_{i+1} or R. If the reply data are simply forwarded, anonymity will be decreased since the data linkage in a transmission channel is easily specified. U finally obtains the multiplexed ciphertext of reply data from R and decrypts it using n session keys between the user and each relay proxy. Furthermore, for the second and subsequent communications that take place in the same session, we specify that since the initially generated session key is reused based on session ID management, additional key generation is unnecessary. That is, as long as this takes place in the same session, the second and subsequent communications can be performed faster than the initial communications.

The concrete processes for anonymous communication and the anonymity revocation have the following characteristics.

- The anonymous communication scheme proposed in [6], which is a hybrid encryption scheme and is superior to other schemes in regard to communication cost, is implemented as a basic anonymous communication scheme.
- $\mathcal{E}_i$ and $\mathcal{P}$ are hybrid encryption schemes based on [6].

Also, some notations and symbols are followed by [6] and Secs. 2 and 3 in this paper, in particular,

- p and q are primes such that $q|p-1$ holds,
- g is an element of $(\mathbb{Z}/p\mathbb{Z})^{\times}$ whose order is q,
- $\langle g\rangle$ is a unique subgroup of $(\mathbb{Z}/p\mathbb{Z})^{\times}$ generated by g,
- $a_i \in \mathbb{Z}/q\mathbb{Z}$ and $h_i \overset{\text{def}}{=} g^{a_i} \bmod p$ are respectively secret and public keys of the i-th proxy,
- $\mathcal{K}$ is the space for keys,
- and $\mathcal{H}$ and $\mathcal{F}$ are hash functions that map $\langle g\rangle \rightarrow \mathcal{K}$ and $\langle g\rangle \rightarrow \mathbb{Z}/q\mathbb{Z}$, respectively.

The procedure for proposed anonymous communication scheme is as follows.

Processing flow during communications in implemented scheme

1. U performs the following.
   (a) After password verification, access P and acquire the address of R, Addr_{m+1}, and the relayed anonymous proxy address, Addr_i $(i = 1, \ldots, m)$.
   (b) Select $r \in \mathbb{Z}/q\mathbb{Z}$, compute $G_{-1} \overset{\text{def}}{=} g^r$, the key exchange public key sent to P, and as $\mathit{\Lambda}_{-1} \overset{\text{def}}{=} 1$ compute $G_i \overset{\text{def}}{=} h_i^{r\mathit{\Lambda}_{i-1}}$, $\mathit{\Lambda}_i = \mathcal{F}(G_i)\mathit{\Lambda}_{i-1}$, and common key $K_i = \mathcal{H}(G_i)$ for $i = 0, 1, \ldots, m, m+1$.
   (c) For message $C_{m+1} \overset{\text{def}}{=} msg$ compute $C'_i = (\mathsf{Addr}_{i+2} \| C_{i+1})$ and ciphertext $C_i = E_{K_{i+1}}(C'_i)$ for $i = m, m-1, \ldots, 0, -1$. Then send (G_{-1}, C_{-1}) to P (Addr_{m+2}:null).
2. P performs the following.
   (a) Compute $T_0 = G_{-1}^{a_0}$, the public key for the key exchange sent to M_1, $G_0 = G_{-1}^{\mathcal{F}(T_0)}$, and the two common keys $K_0 = \mathcal{H}(T_0)$ and $\mathcal{K}_0 = \mathcal{H}(G_0^{a_0})$. Decrypt C_{-1} as $D_{K_0}(C_{-1})$ and obtain $C'_{-1} = (\mathsf{Addr}_1 \| C_0)$.
   (b) Compute the illegal user tracking ciphertext $\mathit{\Gamma}_0 = E_{\mathcal{K}_0}(G_0 \| \mathsf{UID})$ and signature $\sigma_0 = \mathcal{S}_0(G_0 \| C_0 \| \mathit{\Gamma}_0)$, and send M_1 $(G_0, C_0, \mathit{\Gamma}_0, \sigma_0)$ according to Addr_1, where UID represents a unique ID of U.
3. M_i performs the following for $i = 1, \ldots, m$.
   (a) As a signature authentication for σ_{i-1}, after verifying that $\mathcal{V}_{i-1}(G_{i-1} \| C_{i-1} \| \mathit{\Gamma}_{i-1} \| \sigma_{i-1}) = 1$ holds, compute $T_i = G_{i-1}^{a_i}$, the public key for the key exchange sent to M_{i+1} (if $i = m$, then R), $G_i = G_{i-1}^{\mathcal{F}(T_i)}$, and the two common keys $K_i = \mathcal{H}(T_i)$ and $\mathcal{K}_i = \mathcal{H}(G_i^{a_i})$. Decrypt C_{i-1} as $D_{K_i}(C_{i-1})$ and obtain $C'_{i-1} = (\mathsf{Addr}_{i+1} \| C_i)$.
   (b) Compute ciphertext $\mathit{\Gamma}_i \overset{\text{def}}{=} E_{\mathcal{K}_i}(G_{i-1} \| \mathit{\Gamma}_{i-1} \| \sigma_{i-1})$ for tracking the illegal user and signature $\sigma_i = \mathcal{S}_i(G_i \| C_i \| \mathit{\Gamma}_i)$, and send M_{i+1} $(G_i, C_i, \mathit{\Gamma}_i)$ according to Addr_{i+1}.
4. R performs the following.
   (a) Verify that $\mathcal{V}_m(G_m \| C_m \| \mathit{\Gamma}_m \| \sigma_m) = 1$ holds, and compute $T_{m+1} = G_m^{a_{m+1}}$ and common key $K_{m+1} = \mathcal{H}(T_{m+1})$. Decrypt C_m as $D_{K_{m+1}}(C_m)$ and obtain $C'_m = C_{m+1} = msg$.
   (b) Save input $(C_m, \mathit{\Gamma}_m, \sigma_m)$, and execute processing for message $C_m (= msg)$. Compute ciphertext $R_{m+1} \overset{\text{def}}{=} E_{K_{m+1}}(resp)$ for reply data *resp* to U and send M_m response R_{m+1}.
5. For M_i (if $i = 0$, then P) where $i = m, m-1, \ldots, 1, 0$, calculate ciphertext $R_i \overset{\text{def}}{=} E_{K_i}(R_{i+1})$, and reply to M_{i-1} with R_i (if $i = 1$, then P; and if $i = 0$, then U).
6. U decrypts R_0 as $D_{K_{m+1}}(D_{K_m}(\cdots(D_{K_0}(R_0))\cdots)) = resp$, and obtain reply data *resp* from R.

The procedure for proposed anonymous revocation scheme is as follows.

Processing flow of implemented scheme when tracking an illegal user

1. R transmits $\mathsf{LOG} = (G_m, C_m, \Gamma_m, \sigma_m)$ to P, and requests that the users that support the LOG be disclosed.
2. P decides on the need for user disclosure based on some reason or another, and if disclosure is determined, P performs the following.
3. Verify that $\mathcal{V}_m(G_m \| C_m \| \Gamma_m \| \sigma_m) = 1$ holds. If not, R is determined to be illegal and the process is terminated.
4. For $i = m, \ldots, 1, 0$, the following is performed.
   (a) With the cooperation of T_j $(j = 1, \ldots, t)$, compute $\mathcal{K}_i = \mathcal{H}(G_i^{a_i})$, $D_{\mathcal{K}_i}(\Gamma_i) = (G_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1})$, $K_i = \mathcal{H}(G_{i-1}^{a_i})$, and $C_{i-1} = E_{K_i}($ $\mathsf{Addr}_{i+1} \| C_i)$ (if $i = 0$, then $D_{\mathcal{K}_i}(\Gamma_i) = (G_{i-1} \| \mathsf{UID})$).
   (b) If $i > 0$, after verifying the legal processing of T_j based on zero-knowledge proof technology, confirm that $\mathcal{V}_{i-1}(G_{i-1} \| C_{i-1} \| \Gamma_{i-1} \|$ $\sigma_{i-1}) = 1$ holds. If not, M_i is determined to be illegal and the process is terminated.
5. The illegal user is specified from UID.

5.2 Performance

We performed the following measurements.

[**Table 1**] shows the processing time in the proposed anonymous communication process where the number of relay proxies, n, is set to 3, 4, 5, 10, 15, 20, and 30 by setting the received data size to 100 KB.

[**Table 2**] shows the processing time in the proposed anonymous communication process where the received data size is set to 1, 10, 100, and 1000 KB for a two-stage anonymous proxy.

[**Table 3**] shows the processing time in the proposed anonymity revocation process where n is set to 2, 3, 5, 10, 20, and 30 by setting the received data size to 100 KB.

In Tables 1, 2, and 3, the data size of the transmission HTTP request ($= msg$) is set to approximately 260 Bytes. The test environment parameters are given in Table 4. The network comprises 100Base-TX switching hubs.

5.3 Evaluation

Based on the results in Table 1, we confirmed that the processing time for the transceiver increases roughly in proportion to the number of relay proxies. This means every relay proxy requires about the same cost while the processing costs for C_{i-1} decryption and Γ_i encryption vary for each proxy M_i. On the other hand, in the case where the transmission message, msg, or the received data, $resp$, are sufficiently large, we can predict different consequences from the increasing trends shown in Table 1.

Table 1. Required Processing Time For Number of Stages in Anonymous Proxy [sec]

	Number of relay proxies (= n)						
	3	4	5	10	15	20	30
Transmission (U → R)	0.238	0.290	0.343	0.630	0.893	1.169	1.759
Response (R → U)	0.310	0.362	0.388	0.568	0.807	1.035	1.495
Transceiver as a whole	0.549	0.674	0.741	1.213	1.703	2.212	3.261

Table 2. Required Processing Time During Communications [sec]

	Received data size			
	1KB	10KB	100KB	1MB
Transmission (U → R)	0.189	0.189	0.184	0.189
Response (R → U)	0.030	0.041	0.213	2.151
Transceiver as a whole	0.227	0.241	0.430	2.347

Table 3. Processing Time Required to Identify Illegal Person [sec]

Number of relay proxies (= n)					
2	3	5	10	20	30
1.122	1.269	1.547	2.250	3.587	4.828

Table 4. Test Environment

CPU	Pentium 4 3.0 GHz
RAM	1 GB
NIC	1000BASE-T
OS	Windows 2000 Professional SP4
Language	C or Java
Protocol	SOAP over HTTP

From the results in Table 2, even if the size of the received data is around 100 KB, we find that the processing cost for response exceeds that for transmission, which includes the key exchange and ciphertext generation for revoking anonymity. In addition, we verified that even if the size of the received data is around 1 MB with a two-stage relay it takes only around 2 sec. and in terms of viewing as web services, the performance is generally sufficient for practical purposes.

The processing for identifying an illegal user or his accomplice is not generally considered to require immediate promptness compared to when conducting communications; however, as shown in Table 3 even if the number of stages n of relay proxies is large (e.g., 30), it was confirmed that the processing can be accomplished within a few seconds.

6 Conclusion

We proposed a scheme that provides anonymous communications with a function to track malicious users over a communication channel in which monitoring of the channel is intractable. The proposed scheme supports semi-asynchronous and asynchronous modes using Onion Routing. We also reported on the implementation results of the asynchronous mode of proposed scheme to verify the degree of practicality. The results verified that generally the performance is sufficient for real-time web services in which both user anonymity and preventing illegal use are desired such as anonymous bulletin board services, auctions, and peer-to-peer file exchange.

Acknowledgement

We would like to thank the anonymous referees for valuable comments and suggestions to improve the quality of the paper.

References

1. Ateniese, G., Song, D., Tsudik, G.: Quasi-efficient revocation of group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
2. Bresson, E., Stern, J.: Group signature scheme with efficient revocation. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001)
3. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
4. Chaum, D.L.: Untraceable electronic mail, return address, and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
5. Chaum, D.L., Heyst, E.V.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
6. Chida, K., Abe, M.: Flexible-routing anonymous networks using optimal length of cipher text. IEICE Trans, Fundamentals E88-A(1), 211–221 (2005)
7. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
8. Nakanishi, T., Sugiyama, Y.: A group signature scheme with efficient membership revocation for reasonable groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 336–347. Springer, Heidelberg (2004)
9. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: Proc. 1997 IEEE Symposium on Security and Privacy, pp. 44–54. IEEE Press, Los Alamitos (1997)
10. Tsudik, G., Xu, S.: Accumulating composites and improved group signing. In: Laih, C-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (2003)

GAS: Overloading a File Sharing Network as an Anonymizing System

Elias Athanasopoulos[1], Mema Roussopoulos[2,1], Kostas G. Anagnostakis[3], and Evangelos P. Markatos[1]

[1] Institute of Computer Science (ICS)
Foundation for Research & Technology Hellas (FORTH)
{elathan,mema,markatos}@ics.forth.gr
[2] Harvard University
mema@eecs.harvard.edu
[3] Institute for Infocomm Research, Singapore
kostas@i2r.a-star.edu.sg

Abstract. Anonymity is considered as a valuable property as far as everyday transactions in the Internet are concerned. Users care about their privacy and they seek for new ways to keep secret as much as of their personal information from third parties. Anonymizing systems exist nowadays that provide users with the technology, which is able to hide their origin when they use applications such as the World Wide Web or Instant Messaging. However, all these systems are vulnerable to a number of attacks and some of them may collapse under a low strength adversary. In this paper we explore anonymity from a different perspective. Instead of building a new anonymizing system, we try to overload an existing file sharing system, Gnutella, and use it for a different purpose. We develop a technique that transforms Gnutella as an Anonymizing System (GAS) for a single download from the World Wide Web.

Keywords: Security, Anonymity, P2P, Gnutella.

1 Introduction

Anonymity is considered as a valuable property in both physical and digital life. Acting in an anonymous fashion may be considered suspicious, and in many cases the actor may not follow legal procedures. Apart from these ill cases, anonymity is considered the basic building block for someone who demands to keep her privacy. We are interested in cases, in which technology is used to preserve a user's private information from third parties.

David Chaum[7] in 1981 was the first to vision an electronic mail service that will operate in an anonymous fashion, using some intermediate MIX nodes. These intermediate MIX nodes decouple the correlation between the sender and the recipient; this correlation is the outcome of having the recipient contacting directly the electronic mail service. Since then, many systems that are especially developed in order to serve as anonymizing systems have been published.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 365–379, 2007.
© Springer-Verlag Berlin Heidelberg 2007

These constructs may be used as means for having interactions with third parties without revealing the user's identity. More precisely, anonymity is usually required when two parties are having a transaction. For example a Web client is visiting a Web server. We distinguish between *sender* and *receiver* anonymity. In this paper we are dealing with sender anonymity; we develop techniques that allow a user to send an HTTP request to a Web server and receive the Web content, without revealing her identity.

Many anonymizng systems like Crowds[14], Tarzan[10], Cashmere[20] and MorphMix[15] have been designed theoretically and released as academic publications; some others, like Tor[11] and Freenet[8], have been deployed and are actively used. All these systems have been created from the ground up, with the explicit purpose of providing guarantees for a user's anonymity. Using, for example, Tor[11] a user may surf the World Wide Web without revealing her identity (her IP address) to the Web sites she visits.

In this paper, we do not build another anonymizing system. Instead, we use an existing system, Gnutella[1], whose prime goal is the exchange of files between users, as a mean of delivering a file from a Web site to a user in an anonymous fashion. We also present guidelines on how to use Gnutella as a covert communication channel.

The main contributions of this paper are the followings:

- We overload a file-sharing system for anonymous Web downloads.
- We use an existing system in a way, which was not included in the original implementors' intentions.
- We evaluate the strength of anonymity provided by our technique using metrics already proposed by academia.
- We present some ideas and guidelines on how to use the Gnutella system as a covert communication channel.

The rest of this paper is organized as follows. Section 2 highlights the basic technical details of the Gnutella architecture. This Section aims on making the reader, who has not further experience with Gnutella in the past, be able to follow the rest of the paper. In Section 3 we present the fundamental concepts of our techniques, which transform Gnutella into a means for anonymous Web downloads and in Section 4 we evaluate the strength of anonymity provided by Gnutella using well established metrics. In Section 5 we present some ideas on how to transform Gnutella into a covert communication channel. We present related efforts in Section 6 and we conclude in Section 7.

2 Gnutella Architecture

Gnutella[1] is an open system targeting file sharing. It promotes the peer-to-peer paradigm and it is purely a decentralized distributed system having millions of concurrent participants. This section describes fundamental concepts of the Gnutella architecture.

2.1 Generic View

Gnutella is built on a two layer random graph topology. The core layer composed by some thousands of peers that are involved in routing of messages. These peers are also called Ultrapeers. Each Ultrapeer maintains approximately 30 connections with other Ultrapeers and approximately 30 connections with peers of the second layer, which are called Leaves. Leaves are not involved in routing; they send and receive messages via their Ultrapeers. Each Leaf is connected into approximately 3 Ultrapeers.

More information about the Gnutella topology and peer distribution's characteristics can be found in [18].

Each basic operation in the Gnutella system is carried out by constructing messages and routing them through the overlay. The basic lookup operation uses the flooding algorithm to query the overlay. A peer constructs a message that embeds search criteria relative to the file it is looking for, and it forwards the query message to its neighbors. Its neighbors further forward the message to their neighbors and so on. Along the path the original message is routed, every peer is free to answer to the query, by constructing a message with possible results relative to the search criteria and its identity; namely its IP address and a Port number. This message is routed back to the original peer that issued the lookup operation following the reverse of the path taken by the query.

Finally, if a peer is satisfied with the search results, it connects to the peer using the identity embedded in the search results message and it downloads the file using the HTTP protocol.

2.2 Gnutella as a Web Download Platform

It is shown in [5] that with the current Gnutella architecture it is possible for a Web server to become advertised in Gnutella search results messages. Furthermore in [5] the authors have developed techniques that trick Gnutella peers into downloading a specific file from a Web server.

Shortly, in [5] the authors managed to misuse the lookup procedure of the Gnutella protocol, in order to force peers to download files from third parties, and especially, as it is illustrated in detail in [5], from Web servers. The main idea, is to insert a malicious peer in the Gnutella system that answers query messages with responses, which instead of containing the identity of the malicious peer, they contain the identity of a third party; a Web server. This is feasible, because Gnutella is completely decentralized and there is no way to distinguish between a message contains authentic information or a message that is specially crafted to contain fake information.

Acting as described above has the effect of tricking Gnutella peers to try to download files from a Web server. Furthermore, the authors in [5] have managed not only to trick Gnutella peers to request a file from Web server, but they illustrate that it is possible to trick Gnutella peers to *actually download a specific file* from a Web server. This can be achieved by embedding special crafted filenames in query responses that take advantage of some HTTP characteristics, so as to force a Web server to serve a specific file upon accepting a download request from a Gnutella peer.

The authors' intentions in [5] was to cause a distributed denial of service attack to a Web server by tricking Gnutella peers to massively request a file from a Web server. In this paper, we build upon this idea to trick a few peers to fetch a file from a Web server inside Gnutella and then make a user able to download the file from Gnutella instead of requesting it from the Web server. In this way, a user can download a file from a Web server in an anonymous fashion; without coming in direct contact with the Web server.

2.3 HOPs Spoofing

A vital characteristic of the Gnutella system is that you can never tell if a message originates from a peer or the latter simple routes the message on behalf of another one. More precisely, each message of the Gnutella protocol has a TTL (Time To Live) and HOPs field. Each peer is responsible for increasing the HOPs field and decreasing the TTL field upon a route operation. When the TTL field reaches the zero value, the peer that received the message is responsible for dropping the message from the system. The interesting part is that each peer is free to create messages with spoofed TTL and HOPs field. In this fashion, a peer can inject a new message into the system, but, by spoofing the HOPs and TTL field, the peer can pretend that the message is routed and not created by itself.

3 GAS Architecture

In this section we analyze in detail the GAS architecture. In Table 1 we list the meaning of symbols used frequently in the current and following sections.

3.1 Overview

GAS' goal is to transfer a file from a Web Server to a computer machine in an anonymous fashion. Strictly, what we are trying to achieve is:

> **GAS' Goal**: Transfer a file, F, from a Web Server, W, to a computer machine, C, that never comes in a direct contact with W, nor with another computer, which is able to prove with a great probability that C was in contact with W in order to retrieve the particular file.

In order to achieve GAS' goal we use the real-world file-sharing system Gnutella. The whole GAS algorithm is divided into two separate phases: the *FETCH* and the *MIX* phase. The *FETCH* phase transfers F from W to some peers in the Gnutella system and the *MIX* phase populates F in the Gnutella system. We explore the two different phases of GAS in the following paragraphs.

Table 1. Symbols which are used frequently in the description of the GAS architecture and evaluation

Symbol	Explanation
C	Computer, GAS user, the user who desires the anonymity
F	The desired file located in a Web sever
W	The Web server that hosts the desired file
$FETCH$	Initial phase of GAS (a file is transfered from the Web server to Gnutella)
MIX	Second phase of GAS (a file hosted by Gnutella peers is further propagated to the Gnutella system)
d	Degree of Anonymity (anonymity metric)
H_M	Maximum entropy of the system
$H(X)$	Entropy of the system after the adversary's attack
N	Peers that compose the set of the $FETCH$ phase
M_i	Peers that compose the set of the ith MIX phase
A_c	Peers which are controlled by the adversary

3.2 FETCH Phase

The *FETCH* phase is the initial part of GAS. The main goal of the *FETCH* phase is to transfer F from W to Gnutella. This must be done, without having a direct contact of C with W. Thus we use the techniques illustrated in [5]. In [5] the authors present a technique to use Gnutella as a Denial of Service attack platform against Web Servers. More precisely the authors have developed a technique that tricks Gnutella peers to download a file from a third party Web server. If a large population of Gnutella peers is tricked to download from the Web server, the Web server is faced with a flash crowd event and eventually becomes a victim of a distributed Denial of Service attack. In GAS we employee the same idea, but we use it in a controlled way, so as not to cause any harm in Web servers.

GAS uses a special modified peer, C, as it is done in [5], in order to advertise F, which is served by W, in Gnutella, so that some Gnutella peers proceed and download the file.

During the *FETCH* phase we assume that N Gnutella peers will have downloaded F from W. Note, that there is no way - up to this point - for any external observer to associate with any information C with W. Since C is connected to the Gnutella system and feeds it with fake results, that embed the identity of W.

According to the experiments illustrated in [5] a modified peer which issued 10,000 fake results managed to trick 133 Gnutella peers in 10 minutes of working time. More experiments can be found in [5].

Up to this point, we have managed anonymously to force Gnutella to download F from W. We could use C to query for F the Gnutella system, using some keywords that are relevant to F's name and then download F from one peer that belongs to the set of peers N that have fetched F from W. In this fashion, C has managed to retrieve F without coming in direct contact with W but only with some tricked Gnutella peers. However, as we further investigate in Section 4, we assume that an adversary against GAS has managed to control a peer, A, which belongs to the N set of Gnutella peers. This peer may associate C with W since A and W may cooperate in order to attack GAS. In order to deal with this case we employ the *MIX* phase.

3.3 MIX Phase

The *MIX* phase aims on populating F in the Gnutella system in a chaotic way. That is, C tries to further trick other Gnutella peers to download F from the N peers that were tricked during the *FETCH* phase.

In order to achieve this C needs to have a list of IP addresses that potentially map to tricked peers belonging to the N set. As it is explained in detail in [5], in order for a malicious peer to trick a peer to download a file served by a Web Server or another Gnutella peer, the malicious one must know in advance the IP address of the victim, in order to embed it in fake QueryHits (reply messages to Query messages). It is hard for C to know exactly which peers were tricked to download F from W, since the download process via Gnutella involves in great extent the human factor. However, C can try to use as victims all the peers which it sent fake responses during the *FETCH* phase. This process requires C to know for which peer it generates a fake response when it receives a Query message. In the original Gnutella specification a Query message does not embed the identity (IP address and Port number) of the initial querier. During the last few years, an extension has been introduced in order to promote direct delivery of responses to requesting peers via UDP[4]. Peers that support this extension include their identity in their Query messages. We performed measurements, using a modern Gnutella client[2], in order to find out the ratio of popularity of this extension in current Gnutella. In other words, we measured how many Query messages embed Gnutella identities. In Table 2 we list the results. The majority of Query messages in current Gnutella embed identities of Gnutella peers. Thus, C can generate fake responses only for Query messages, which embed a Gnutella identity and keep the identity in a list. These collected identities may potentially take part in the N set.

Table 2. OOB Queries percentage, measured using a modern Gnutella client [2]. OOB Query messages embed the identity (IP Address and Port number) of the peer that issued the Query.

Queries Received	OOB Queries	Percentage
1,000	751	75%
2,000	1,536	77%
3,000	2,183	73%
4,000	2,840	71%
5,000	3,498	77%
10,000	6,617	66%

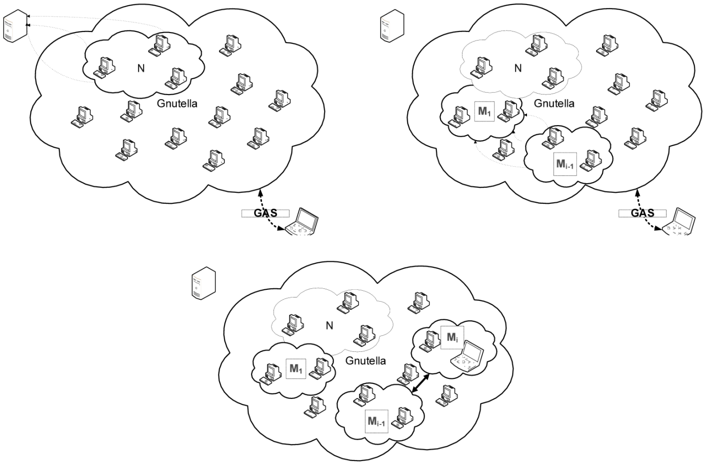


Fig. 1. The different phases of the GAS architecture. During the *FETCH* phase N Gnutella peers are tricked to download a file from the Web Server. During the *MIX* phases, the file is further populated in the Gnutella system (in M_i sets). Finally, the user of GAS participates in the final M_i set and downloads the requested file from peers in the M_{i-1} set.

Having collected an identity list of potentially tricked peers, the N set, C may proceed and further trick other Gnutella peers to download F from peers belonging to N set. These new tricked peers compose the M_1 set.

The *MIX* phase may be applied i times in order to gain stronger anonymity. Finally, C may query the peers composing the M_{i-1} set and download F from them. That is C will be - artificially - part of the final M_i set.

4 GAS Evaluation

There are several attempts to quantify the anonymity provided by an anonymizing system. Most recent papers [9], [16] and [19] use the notion of information entropy to measure the information which leaks from the system and can be used by an adversary in order to degrade the anonymity provided by the system. We will use [9] in order to measure the *degree of anonymity* provided by GAS. This theoretic metric depicts the effort required from an adversary to be able to identify with great probability a sender from a set of possible senders.

In [9] the authors have illustrated the degree of anonymity of known systems such as Crowds[14] and Onion Routing[11] and, thus, using the same methodology GAS can be compared with the above systems.

Before we proceed, in the rest of this evaluation the term *anonymity* follows the precise definition given by [13]: *the state of being not identifiable within a set of subjects, the anonymity set.*

The degree of anonymity as expressed in [9] is related to *sender anonymity*. Of course, as the authors mention in the paper, *recipient anonymity* can also be modeled using the same concept. It is vital to understand that, in contrast to other anonymizing systems, GAS introduces the notion of separate phases, the FETCH and a series of possible MIX phases. In order for complete Web transaction to take place using GAS, C becomes a *sender* and a *recipient*. It is a pure sender during the FETCH phase and the possible MIX phases, but in the final stage, when the actual download is performed from the M_i set, C is a recipient. Thus, we have to evaluate the degree of anonymity of GAS in two separated phases: one, in which the user acts like a sender and one, in which the user acts like a recipient.

4.1 Degree of Anonymity

As the authors state in [9] the degree of anonymity is expressed as:

$$d = \frac{H(X)}{H_M}, \tag{1}$$

where

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

is denoted as the entropy of the system after the attack has taken place, while

$$H_M = \log_2(N),$$

is denoted as the maximum entropy of the system, when N is the size of the anonymity set that includes the number of legitimate senders (or recipients).

In the above, we denote with p_i the probability mass function $p_i = Pr(X = i)$, where i represents each possible value the discrete random value X may take. Specifically, each i corresponds to an element of the anonymity set (a sender or a recipient). If we have a system with i possible senders, an adversary assigns a probability p_i to each one of the set.

4.2 Degree of Anonymity of GAS

In order to estimate the degree of anonymity in GAS we have to define the adversary model. We believe that a realistic adversary model is an *active attack*. That is, the attacker has under its full control A_c nodes that take part in the GAS *MIX* phase.

The maximum entropy of GAS can be measured if we define the anonymity set that includes the nodes of Gnutella that take part in the process of the file downloading. Let G_p to be the complete population of Gnutella nodes. The nodes that take part in GAS are

$$N + \sum_{i=1}^{m} M_i,$$

assuming we have m *MIX* phases. It is always

$$N + \sum_{i=1}^{m} M_i \leq G_p.$$

The maximum entropy of GAS is:

$$H_M = \log_2(N + \sum_{i=1}^{m} M_i). \tag{2}$$

Now, each node in the A_c may come in contact with:

- the Web server, W, and possibly with a Gnutella peer that belongs to the M_1 set, if it is part of the N set,
- another Gnutella peer that belongs to M_{i+1} if it is part in the M_i set.

The adversary must assign probabilities to each node it comes in contact with. Assuming, that the GAS user is hidden in a Gnutella subset, i.e. it may not be distinguished in a trivial way from other Gnutella nodes that take part in an M_i set, the probability is uniform and depends on the requests recorded from the adversary. Thus, the adversary assigns $p_i = \frac{1}{R_{in,i}}$, where $R_{in,i}$ denotes the number of incoming requests of a node controlled by the adversary which is in an M_i set. It is always $R_{in,i} \leq M_{i+1}$.

There is only one case, where the adversary may spot the GAS user with great probability. Assuming we have m *MIX* phases, if the adversary manages to control all nodes in M_{m-1} set and the final set M_m includes only the GAS user, then the identity of the GAS user is fully revealed. We argue that, although this is possible, it may be adjusted by the GAS user, so that each M_i set is quite dense. This is manageable, since the GAS user constructs the sets, by tricking other Gnutella peers. In addition, it is difficult for the adversary to know in which *MIX* set she belongs to - it requires the effort of having at least one node controlled by the adversary in each *MIX* set.

Thus, in the general case and assuming that $R_{in,i} = M_{i+1} - A_c$ (this, simply, means that a node in the M_{i+1} set is forced to generate requests to each node of the M_i set, excluding nodes controlled by the adversary), the probability assigned by the attacker to each GAS participant is $p_i = \frac{1}{M_{i+1} - A_c}$. The entropy for a given set k will be:

$$H(X) = - \sum_{i=1}^{M_{k+1}-A_c} \frac{1}{M_{k+1} - A_c} \log_2(\frac{1}{M_{k+1} - A_c}). \tag{3}$$

In order to calculate the entropy of the whole system, we need to sum up all the probabilities for all *MIX* phases.

4.3 Degree of Anonymity in a *MIX* Phase

Assuming we have only one *MIX* phase the degree of anonymity is:

$$d = - \frac{\sum_{i=1}^{M-A_c} \frac{1}{M-A_c} \log_2(\frac{1}{M-A_c})}{log_2(M)} = \frac{log_2(M - A_c)}{log_2 M}. \tag{4}$$

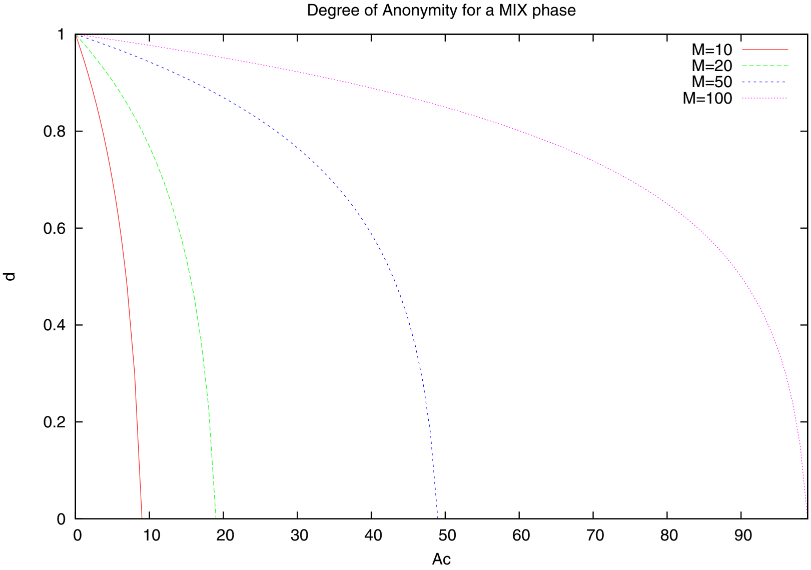


Fig. 2. Degree of Anonymity for a MIX phase for various set's populations. The degree of anonymity is expressed in equation (4). Initially the degree of anonymity provided by the system is equal to 1 (complete anonymity). As the adversary injects more nodes in the system, the degree of anonymity is reduced. If the adversary manages to substitute all nodes with nodes controlled by her, then the anonymity degrades to zero.

Since, we are interested in the degree of anonymity of a *MIX* phase, we excluded from H_M the N nodes that take part in the initial *FETCH* phase.

In Figure 2 we depict the degree of anonymity for a given *MIX* phase for various set's populations.

4.4 Degree of Anonymity of a Series of MIX Phases

In order to calculate the degree of anonymity of the whole system, we have to sum up all the MIX sets and extend equation (4) to include the nodes that compose all the MIX phases. It is trivial to observe that equation (4) will not differ with the equivalent of a series of MIX phases, since in order to calculate the entropy of a series of MIX phases we just sum up all senders (all the M_i sets) and subtract the nodes which are controlled by the adversary. Thus:

$$d = -\frac{\sum_{i=1}^{M_k-A_c} \frac{1}{M_k-A_c} \log_2(\frac{1}{M_k-A_c})}{log_2(M_k)} = \frac{log_2(M_k - A_c)}{log_2 M_k}, \tag{5}$$

where now with M_k we denote the sum of the nodes of all the MIX phases. We omit the $\sum$ symbol for readability reasons.

4.5 The Benefit of Having Multiple MIX Phases

Comparing equations (4) and (5) reveals that there is no benefit of having multiple MIX phases in terms of the *degree of anonymity* metric. In other words, the degree of anonymity, according to the above model, depends only in the percentage of nodes controlled by the adversary over the nodes that are tricked by the GAS user and serve as possible senders. To give it with numbers, two MIX phases, M_1 and M_2, with A_{c1} and A_{c2} nodes controlled by the adversary have the same degree of anonymity with one MIX phase, M', having A'_c nodes controlled by the adversary, if $M' = M_1 + M_2$ and $A'_c = A_{c1} + A_{c2}$.

This seems a little bit contradictory. Why, then do we need multiple MIX phases? The answer is that having multiple MIX phases forces the attacker to spread the controlled nodes to all MIX sets if she wants to calculate correctly the degree of anonymity of the last MIX set. Recall that the GAS user performs a download - and this action is the only action which qualifies as evidence that someone is utilizing GAS - only as part of the last MIX set.

By having only one MIX phase, the adversary may assign p_i probabilities to all the nodes taking part in the MIX phase. By having multiple MIX phases, the adversary must distinguish nodes that take part in different MIX phases and focus on the nodes of the last MIX set. This requires the effort of having at least one attacker in each of the MIX sets.

The intuition of having multiple MIX phases is the same as having multiple chaumian $MIXes$ in a $MIXnet$ [7].

4.6 Comparison of GAS and Other ASs

In [9] there is a similar evaluation with the above one regarding the degree of anonymity in existing anonymizing systems. It is interesting to observe that the degree of anonymity of GAS is equal to the degree of anonymity provided by Onion Routing [11]. However, there are some properties of GAS that differentiate [1] it from Onion Routing. In terms of advantages, GAS has the following properties:

- The sender sets of GAS can have much more entries than a typical sender set of Onion Routing,
- The adversary must inject many nodes to control a substantial number of nodes in the system, so as to be able to perform a realistic attack.

In terms of disadvantages:

- GAS does not have as fine-grained controls control as Onion Routing,
- GAS can not be used for real-time Web Browsing, but for a simple download from WWW,
- GAS uses unencrypted communications.

[1] The degree of anonymity is a well defined and accepted metric in order to measure the anonymity provided by a system. However, some practical characteristics of a system make it ideal for performing some tasks in an anonymous fashion than another one, even if both systems have exactly the same degree of anonymity.

5 Using GAS as a Covert Communication Channel

Besides using Gnutella as a platform for anonymous downloads via the WWW, Gnutella may also transformed into a covert communication channel. For example, two parties that want to communicate anonymously via Gnutella can exchange Query and QueryHit messages. These messages can be also encrypted. Below we list a numerous ways of using some of the Gnutella internals to transform the system into a covert communication channel.

5.1 Using the Gnutella Lookup Procedure

A peer may flood a query with encrypted payload[2] in Gnutella. A colluding peer, which is located in the initial peer's horizon may answer the Query with a response with encrypted payload[3]. The two colluding peers have managed to:

- Exchange information using the lookup mechanism of Gnutella.
- Exchange information using encrypted messages.
- Exchange information without coming in direct contact with each other, but using Gnutella as a means to route the information from one party to the other.

The Query message, since it is flooded in the system, will be received from many peers, as well as the response in the Query message will be routed back to the initial peer through a series of Gnutella peers. Since both payloads of the Query and the Query response are encrypted, it is unlikely that an adversary that has injected its nodes in the system can perform a Man in the Middle attack and reveal the contents of the messages. The two colluding peers may also exchange secret keys occasionally again using the lookup operation of the Gnutella system.

5.2 Using the Gnutella Message Tagging

Apart from taking advantage of Gnutella's lookup procedure to exchange private information there are more elaborate methods for using the Gnutella system as a covert channel. A lot of Gnutella messages contain unused bits; bits that will never used by a normal Gnutella client. Two colluding peers may inject the information they want to exchange in these unused bits. In order to illustrate a more concrete example, consider that each Gnutella message is tagged with a 16-byte GUID[3]. Assuming that a GUID tag is totally random, two colluding parties may arrange to use some part of the 128-bit GUID tag as an information carrier. All normal Gnutella clients will treat this portion of the GUID tag as a random subportion of the complete identifier, but the colluding peers will treat specially this portion of information according to their prior arrangement.

[2] We consider the search criteria of a Query message as the payload of a Gnutella query.

[3] We consider the search results of a Query response message as the payload of a Gnutella Query hit.

5.3 Using the Gnutella PING-PONG Exchange

A less efficient, in terms of flexibility, technique is to use the PING-PONG exchange mechanism to transmit and receive secret information. Although, this mechanism is not widely used in the current Gnutella system, since it produces network traffic overhead, it is still possible for peers to use this mechanism to discover new hosts. When a peer does not have many entries in its host cache and it is in need of discovering new Gnutella peers, it broadcasts (using again the flooding process) a Gnutella PING packet. In response, it receives Gnutella PONGs from active Gnutella peers. These PONG messages embed the peers' identities (IP address and Port number) as well as some statistics in regards to the data files they share. Again, two colluding peers may encode secret information in these PING-PONG messages. For example a peer may encode the message it wants to transmit in a PONG packet. Peers that passively monitor the PONG traffic will try to connect to a non existing IP address and Port number, since it is unlikely that the special encoded message will map to a valid Gnutella IP address and Port.

The profound property of Gnutella that makes all the above easily achievable is the lack of a central mechanism to verify if a message is authentic or fake. Since every Gnutella participant may inject messages in the system that seem to originate from another node and not by itself, then it is feasible to also embed in these messages secret information, which can only be interpreted in a meaningful way by another colluding node (see HOPs spoofing in Section 2.3).

6 Related Work

Since David Chaum introduced the term of an anonymous and untraceable electronic mail in 1981 [7] a lot of research has been taken place in the academic and industrial field. Nowadays, there are plenty of anonymzing systems, such as Crowds [14], Tarzan [10], MorphMix [15], Freenet [8], Cashmere [20] and Tor [11]. Each system tries to provide anonymous communication by routing messages between a sender and a receiver through nodes that try to decouple any relation between the two communicating parties. It is worth mentioning that all these systems have certain advantages and disadvantages, and thus this is an active field for further research.

As far as GAS is concerned the two key properties that differentiate it from current anonymizng systems are that (a) GAS is built over an existing infrastructure that was not designed to provide any mean of anonymous communication and (b) GAS is built over a large set of nodes that can be used as relayers in order to provide anonymous communication.

As far as the first property is concerned, there has been an attempt for utilizing the World Wide Web as a covert channel [6] in order to provide anonymous communication. Beyond that, there is no similar work in exploiting existing systems, designed for other purposes than anonymous communications, for anonymity purposes, as far as we know.

As far as the second property is concerned, most of current anonymizing systems rely on a small set of nodes that in most cases [10,11,15,20] full knowledge of the system is required, and thus there are scalability issues. Anonymizing systems such as P^5 [17] and Salsa [12] have been designed in order to provide scalable anonymous communications. But, again it is hard to implement a dedicated system for anonymous communication that hosts a few millions of nodes.

7 Concluding Remarks

In this paper we presented GAS (Gnutella as Anonymizing System), which transforms the open file sharing system, Gnutella, to a platform for performing file downloads from the World Wide Web in an anonymous fashion.

We furthermore evaluated GAS with already accepted scientific metrics, such as the degree of anonymity [9]. We compared GAS using the metric of degree of anonymity with other anonymizing systems and showed that GAS has the same degree of anonymity with Onion Routing. We presented crucial properties of GAS which differentiate it from Onion Routing in various aspects.

To conclude, we believe that this paper is the first attempt to use an already existing system, which has not been initially designed as an anonymizing system, for anonymous communication. The fact that GAS is based in Gnutella, which has not been designed for anonymity purposes, makes it less controllable and unsuitable for real-time communications compared to other practical implementation of anonymizing systems, like Tor [11] for example. But, on the other hand, the increasingly popularity of Gnutella, gives GAS, potentially, a very large anonymity set composed by millions of nodes, which, as far as we know, has never been accomplished by any other anonymizing system.

In terms of the degree of anonymity metric and according to equation (5) having a large anonymity set requires from the attacker to inject more attacking nodes in the set in order to reduce the metric, and thus degrade the anonymity provided.

We believe that GAS is not suitable for real-time Web Surfing, but for a single Web download, it is up to the GAS user to trick a huge base of Gnutella peers and form a huge anonymity set that will hide her activities inside.

Last but not least, we listed some possible usages of Gnutella as a covert communication channel.

Acknowledgments

We thank the anonymous reviewers and George Danezis (K.U.Leuven) for his valuable feedback in various aspects of GAS. This work was supported in part by project SecSPeer (GGET USA-031), funded in part by the Greek Secretariat for Research and Technology and by the CoreGrid Network of Excellence. Elias Athanasopoulos, Mema Roussopoulos and Evangelos P. Markatos are also with the University of Crete.

References

1. Gnutella protocol. http://rfc-gnutella.sourceforge.net/
2. Gtk-gnutella servent. http://gtk-gnutella.sourceforge.net
3. GUID Specification. http://en.wikipedia.org/wiki/Guid
4. OOB Specification. http://gtk-gnutella.asselman.com/gtk-gnutella-current/doc/gnutella/out-of-ba nd
5. Athanasopoulos, E., Anagnostakis, K.G., Markatos, E.P.: Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network That Never Forgets. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 130–145. Springer, Heidelberg (2006)
6. Bauer, M.: New Covert Channels in HTTP: Adding Unwitting Web Browsers to Anonymity Sets. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003), Washington, DC, USA (October 2003)
7. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 4(2) (February 1981)
8. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pp. 46–66 (July 2000)
9. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, Springer, Heidelberg (2003)
10. Freedman, M.J., Morris, R.: Tarzan: A Peer-to-Peer Anonymizing Network Layer. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC (November 2002)
11. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding Routing Information. In: Anderson, R. (ed.) Proceedings of Information Hiding: First International Workshop. LNCS, vol. 1174, pp. 137–150. Springer, Heidelberg (1996)
12. Nambiar, A., Wright, M.: Salsa: A Structured Approach to Large-Scale Anonymity. In: Proceedings of CCS 2006 (October 2006)
13. Pfitzmann, A., Hansen, M.:Anonymity, Unobservability, and Pseudonymity: A Consolidated Proposal for Terminology(Draft)(July 2000)
14. Reiter, M., Rubin, A.: Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security 1(1) (June 1998)
15. Rennhard, M., Plattner, B.: Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002), Washington, DC, USA (November 2002)
16. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, Springer, Heidelberg (2003)
17. Sherwood, R., Bhattacharjee, B., Srinivasan, A.: P5: A protocol for scalable anonymous communication. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy (May 2002)
18. Stutzbach, D., Rejaie, R.: Characterizing the two-tier gnutella topology. SIGMETRICS Perform. Eval. Rev. 33(1), 402–403 (2005)
19. Tóth, G., Hornák, Z., Vajda, F.: Measuring anonymity revisited. In: Liimatainen, S., Virtanen, T. (eds.) Proceedings of the Ninth Nordic Workshop on Secure IT Systems, Espoo, Finland, pp. 85–90 (November 2004)
20. Zhuang, L., Zhou, F., Zhao, B., Rowstron, A.: Cashmere: Resilient Anonymous Routing. In: Proc. of NSDI, Boston, MA, (ACM/USENIX) (May 2005)

A Composite Privacy Protection Model

Yi Ren[1,2], Min Luo[1], Zukai Tang[1], and Lingqing Ye[1]

[1] The State-Key Lab. of Software Engineering, Wuhan University, WuHan, Hubei, China
[2] Network Management Center, Communication and Commanding Academy,
WuHan, Hubei, China
rwxing_zh@yahoo.com.cn, robin6925@hotmail.com,
tangzukai@whut.edu.cn

Abstract. The current privacy-preserving researches are based on the relational data model. However, the existing privacy data models based on the relational model exist some shortcomings. First, they are not enough for protection of composite privacy object. The current researches focus on the privacy data of individual and do not consider how to protect the relationship among several privacy objects. The relationship is also a type of privacy data of each individual. Second, the rapid increasing view make it is difficult for the privacy database administrator to manage the privacy database effectively. In this paper, a privacy data model based on deputy mechanism is proposed for solving those problems. The model can depict the generalization relationship among different privacy objects and provide a stronger hiding capability that IS-A relationship. Moreover, compared with object-oriented data model, the model pays more attention to the storage and usage of privacy data objects in the context of database. Finally, this paper illustrates how to implement the privacy model in the object deputy database management system.

Keywords: Privacy Data Model, Privacy Database, Privacy Protection, Deputy Mechanism.

1 Introduction

Privacy is the right of individuals to determine for themselves when, how and to what extent information about them is communicated to others [1, 14]. The current privacy-preserving researches are based on the relational privacy data model [1, 2, 3, 4, 5, 6, 7, 10, 12]. In the privacy data models, for each (*purpose, recipient*) pair, a view of each table is defined. The prohibited cells in each view are replaced with *null* values. The privacy database administrator can limit disclosure of privacy data by limiting the views that can be accessed by a special user/role.

The relational privacy data model is mature and simple, but it exist some shortcomings. First, it is difficult to protect the composite privacy object. A composite object comprises several individual objects and relationship among these objects. So the relationship is also a type of privacy data and must be protected by a privacy database. However, the relationship is not considered in the current relational privacy data model. Second, the number of view in current privacy data model is the product

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 380–395, 2007.
© Springer-Verlag Berlin Heidelberg 2007

of the number of *purpose* and the number of *recipient*. With the addition of role and purpose, the rapid increasing view make it is difficult for the privacy database administrator to manage the privacy database effectively.

The main contribution of this paper is to propose a new privacy data model based on object deputy mechanism (ODM) [8,9] to support the protection of the composite privacy data. Furthermore, the model can reduce the number of view in the privacy database. In our model, the number of deputy class (see section 3.1) is linear growth along the number of the purpose. We can address this issue by the extended ODM model, which substitutes a switch operation set for a switch operation when a deputy class is defined. Such extended ODM model and its implementation is also a contribution of this paper.

The rest of this paper is organized as follows. Section 2 discusses previous related works. Section 3 introduces the deputy mechanism. Section 4 presents the privacy data model based on deputy mechanism. We illustrate how to implement the privacy model in the object deputy database management system in section 5. Section 6 compares our model with the view-based privacy data model. Finally, we propose the future research intention in section 7.

2 Related Works

To the best of our knowledge, the first approach to incorporate privacy protection within relational database system was proposed by Agrawal et. al [1]. The paper has introduced the concept of Hippocratic databases by incorporating privacy protection in relational database systems. The fundamental principles underlying Hippocratic databases and a reference architecture was proposed in this paper. The paper laid the foundation for the subsequent researches of privacy database.

Ji-Won Byun et al. discussed an aspect of privacy protection in [2]. They believed that a delicate balance between an individual's privacy and convenience and data usability by enterprises and organizations is necessary. In [2], the notion of micro-view was proposed and a "NAVIE" model based on micro-view was proposed to show some of capabilities that a suitable privacy management model should provide.

The use of data generation can significantly protect the uncertainty aspect of privacy data. The various data generation techniques were proposed in [13]. The other aspect of privacy data, indistinguishability, was discussed in [3, 4]. And in [5], Xiaokui Xiao et al. discussed a new generalization framework based on the concept of *personalized anonymity*. The solutions were limited only to the static dataset release. An approach to securely anonymizing a continuously growing dataset was discussed in [7].

In addition, an access control method was proposed by Ji-Won Byun et al. in [6]. It is based on the notion of purpose. The model can support a more sophisticated purpose model, which has hierarchical structures and is characterized by several semantic relationships. In the paper, the conventional RBAC model was extended with the notion of conditional role. Another data model for enforcing the limited disclosure rules was proposed in [12], where the limited disclosure rules were stored as a specified privacy policy inside the database and privacy enforcement was based on views, which were constructed based on the rules.

3 Preliminary

Basic concepts and properties of ODM are summarized in this section (for detail, see [8, 9]).

3.1 Deputy Mechanism

Deputy is an inheritance mechanism. It is based on links from deputy objects to their source objects and restricted by switching operation (see section 3.2). If the switching operations are not defined for some attributes and methods of source classes, the deputy classes cannot inherit them. As shown in Figure 1, in order to inherit b of C1, c of C2 and d of C3, the switching operations for b of C1, c of C2 and d of C3 need to be respectively defined by the deputy class C. However, the switching operations for b, c and d cannot be applied to all of the deputy objects defined the deputy class. For example, the switching operations for b of C1 cannot be applied to deputy objects of instances of C2 and C3. In order to solve this problem, a deputy class can be defined with a uniform switching operation that realizes inheritance in such a way. When a deputy object receives a message, if the methods of the deputy class to which it belongs cannot execute the message, the message id forwards to the source class. The method for executing the message will be looked up in the source class. This is similar to the realization of IS-A inheritance but it is based on the links from deputy objects to source objects and their source objects is not the IS-A relationship from the subclass to the superclass. Deputy inheritance mechanism can realize delegation at class level.

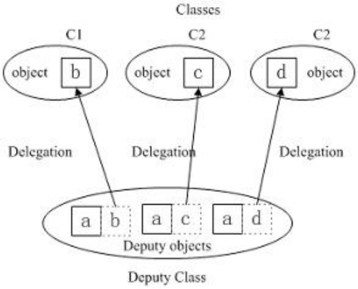


Fig. 1. Atomic privacy object model

3.2 Switching Operation

The switching operation is a key concept in ODM. It is used to define the relationship between an inherited attribute of a deputy object and the corresponding attribute(s) of the corresponding source object(s). Switching operation has two functions.

(1) Controlling inheritance between deputy object (class) and source object (class). If a switching operation is not defined for some attribute or method of source classes, the deputy classes cannot inherit it.

(2) Maintaining consistencies between deputy objects and their source objects. Once a switching operation is defined, the relationship between source object (class) and deputy object (class) will be automatically maintained by ODM. Addition or deletion of an object and modification of attribute value of an object may result in object propagation. The purpose of propagation is to keep relationship between a class and its deputy class. (For more detail about object propagation, see [8,9])

3.3 Deputy Class and Deputy Algebra

In ODM, the real world entities are abstracted as objects and objects with common attributes and methods are grouped into one class. If class p is deputy class of class q, then q is called source class of p. An object can have more deputy objects that represent its multi-faced nature. The deputy object and its source object(s) are connected with bi-directional pointers. Deputy objects can define their own deputy objects as well. A deputy class defines the schema of deputy objects that are derived from source objects.

A deputy class is created by deputy algebra. The deputy algebra contains five types of deputy operations, Select, Union, Join, Project, and Extend. The first three operations work in class level, the last two operations work in attribute level. The operations are used to derive deputy classes for specialization, generalization, extension, aggregation, etc. For example, specialization can be implemented by combining Select, Project, and Extend operations; generalization can be implemented by using operations Union, Project and Extend; and aggregation can be implemented by using operations Join, Project and Extend. Three types of deputy classes, Select, Union, and Join, can be created by these operations. The instances in Select deputy class consist of the instances that are selected from corresponding source class according to selection predication. The instances in Union deputy class consist of the instances of more than one source classes. The instances in Join deputy class consist of the instances that are derived from aggregating instances of corresponding source classes according to combination predication. The formal definitions of these deputy classes can be found in [8]. These deputy classes can be further manipulated by the deputy algebra. This means that ODM possesses a property known as *closure*. It is well known that relational data model has great flexibility because of the closure of relational algebra.

4 The Privacy Data Model Based on Deputy Mechanism

The privacy data model based on deputy mechanism consists of two components: (1) the atomic privacy object model; (2) the composite privacy object model. The former is used to protect privacy of individual, and the latter is used to protect privacy among individuals. The privacy data can be categorized as follows.

(1) Atomic privacy object. It is a single reference signifying a single individual (for example, Tom is ill).
(2) Composite privacy object. It is several references signifying individuals (for example, John is the doctor of Tom). Composite privacy object can be created

by combining more atomic privacy data according to some special combination semantic. Its complexity depends on the complexity of combination semantic. Once a composite privacy object is created, we can treat it as an atomic privacy object.

4.1 Privacy Policy

A privacy policy and a privacy preference are the base of privacy protection. In our model, privacy policies represent the data administrator's policies for data collection and usage; privacy preferences represent the data provider's intention. A privacy policy or a privacy preference is represented as a policy object, which is stored in a basic class, called *privacy-policy*.

Definition 1. Both the privacy policy and the privacy preference are defined as a basic class that includes at least six attributes. Its schema is defined as follows.
BasicClass *privacy-policy* :
:Attribute
(*purpose* : string) // a usage purpose
(*purposetype* : string) // purpose type, "positive" or "negative"
(*classname* : string) // class name
(*attributename* : string) //attribute name of *classname*
(*externalrecipient* : set-of string) //a set of external recipients
(*retention* : DataTime) //a retention period indicating how long *externalrecipient* can use *attributename*

The *privacy-policy* class is created when the privacy database is initialized. An instance of *privacy-policy* class indicates a rule that the attribute of a class must comply. A positive purpose indicates some requisite attributes for the purpose, whereas a negative purpose indicates some attributes that cannot be accessed for the purpose. An example of privacy policy for *payment* purpose on class *patient* in a hospital system is defined in Table 1. The policies in table1 allow the *Accountant* role to access the *Name*, *CreditCard* and *Cost* attributes and prohibit anyone to access the *Address* attribute for *payment* purpose.

Table 1. An example of privacy policy

Purpose	Purpose Type	Class Name	Attribute Name	Externalrecipient	retention
payment	positive	patient	Name	{Accountant}	2007-1-7
payment	positive	patient	CreditCard	{Accountant }	2007-1-7
payment	positive	patient	Cost	{Accountant }	2007-1-7
payment	negative	patient	Address		

Once privacy policies for a purpose are defined, a class hierarchy of privacy data will be created by system automatically. We will discuss how to create the class hierarchy of privacy data in section 4.2.

4.2 Atomic Privacy Object Model

In the real world, individual privacy information is abstracted as privacy objects, which are stored in privacy classes. The individual privacy information is categorized into three groups according to the intention of data owner. The first one is the privacy information that the owner doesn't expect to disclose for the purpose. The second one is the privacy information that must be known by others for completing the purpose. The third one is the privacy information that may be known by others with the explicit consent of the owner. The atomic privacy object model is used to represent the intention of data owner.

The atomic privacy object model is composed of a class hierarchy for privacy data shown in Figure 2. In the class hierarchy, for each purpose, a class sub-hierarchy is defined and consists of four Select deputy classes. The atomic privacy information is grouped as follows. In Figure 2, each sw_{ij} represents a special switching operation and each SW_i represents a switching operation set.

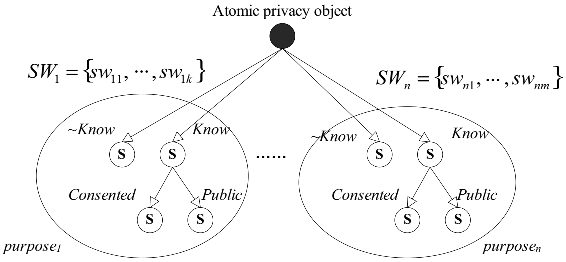


Fig. 2. Atomic privacy object model

(1) *~Know* : It is a set of atomic privacy data of an individual. The objects of the deputy class consist of the privacy data that individuals reject to disclose for the special purpose. It can be used to support negative privacy policy. For example, the "Address" attribute in the table 1.

(2) *Know* : It is a set of atomic privacy data of an individual. The objects of the deputy class consist of the privacy data objects that can be known by other individuals or organization for some special purpose (e.g., purpose$_i$). It is grouped into two following categorizes:

(2-1) *Public* : It is a subset of *Know*. Other individuals or organizations must know the privacy data for achieving the special purpose. It is used to support positive privacy policy. For example, the "Name", "CreditCard" and "Cost" attributes in the table 1.

(2-2) *Consented* : It is a subset of *Know*. Other individuals or organizations can access these privacy data for some special purpose if the data provider gives his/her consent. For example, the attributes that are contained in patient class and are not listed in the table 1.

The formal definition of above 4 types of deputy class is shown in Definition 2.

Definition 2. *~Know*, *Know*, *Consented*, and *Public* deputy class is defined as follows, respectively.

Let *PC*=<O^s, A^s, M^s> denotes source privacy class that stores the privacy data objects, then

(1) *~Know* = <O^d, A^d, M^d>

$O^d = \{ o^d \mid o^d \rightarrow o^s, o^s \in O^s \wedge A^d \in A^s \wedge A^d$ is restricted by negative privacy policy defined in *privacy-policy* class $\wedge$ $SC(o^s) == TRUE$ }, where $o^d \rightarrow o^s$ denotes that o^s is source object of o^d , $SC(o^s) == TRUE$ denote the constraint that o^s must satisfy;

(2) *Know* = <O^d, A^d, M^d>

$O^d = \{ o^d \mid o^d \rightarrow o^s, o^s \in O^s \wedge A^d \in A^s \wedge A^d$ is not restricted by negative privacy policy defined in privacy-policy class $\wedge$ $SC(o^s) == TRUE$ }

(3) *Consented* = <O^d, A^d, M^d>

$O^d = \{ o^d \mid o^d \rightarrow o^s, o^s \in O^s \wedge A^d \in A^s \wedge A^d$ is not restricted by both negative privacy policy and positive privacy policy defined in privacy-policy class $\wedge$ $SC(o^s) == TRUE$ }

(4) *Public* = <O^d, A^d, M^d>

$O^d = \{ o^d \mid o^d \rightarrow o^s, o^s \in O^s \wedge A^d \in A^s \wedge A^d$ is restricted by positive privacy policy defined in privacy-policy class $\wedge$ $SC(o^s) == TRUE$ }

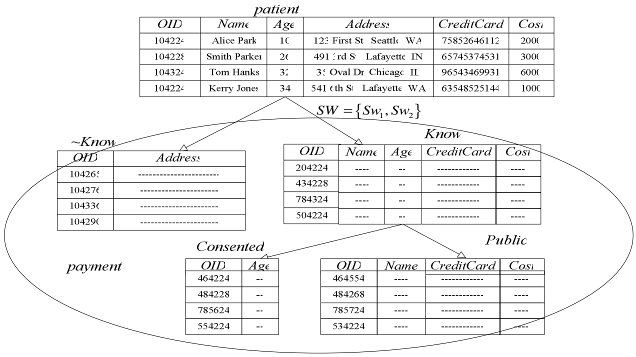
Here, O^s is the set of all the instances of source privacy class, A^s is the set of attributes of source privacy class and M^s is the set of methods of source privacy class. And O^d is the set of all the instances of deputy privacy class, A^d is the set of attributes of deputy privacy class and M^d is the set of methods of deputy privacy class. The constraint $SC(o^s)$, which is represented by a deputy rule, must be specified when the corresponding deputy class is created. So the tuple-level privacy-preserving is implemented by selecting the objects, which satisfying the constraint $SC(o^s)$, from the source privacy class; the attribute-level privacy-preserving is supported by the Select and the Project deputy algebra.

A data user, who can access privacy data for a special purpose, can access the *Public* deputy class of the purpose. If the user wants to access the data in *Consented* class, he/she must get the data owner's consent. For example, if a user *u* is allowed to access privacy data for purpose$_1$, he is granted automatically the permission that can access the *Public* class of purpose$_1$. But if he wants to access *Consented* class, the consent of data owner is required. In order to obtain the consent, several interactions between the data user and database system may be required.

In order to enforce the usability of privacy data, we expand the object deputy model with the notion of switching operation set (*SOS*). As shown in Figure 2, a switching operation set, rather than a switching operation, is defined when *Know* deputy class is created. The data provider will assign a special switching operation to a special user/role when the switching operation is defined. The assignment will be stored in a system schema. The issue will be further discussed in Section 5.

Next, we illustrate our model using the following example. In a hospital system, we consider mainly three data types, *Name*, *CreditCard* and *Cost*, and two user roles, *Info_Admin* and *Accountant.*

According to *privacy-policy* class shown in Table 1, it can be known that the attributes, *Name, Cost,* and *CreditCard*, of *patient* class are requisites for completing *payment* purpose, and the *Address* attribute of patient class does not need to be known for *payment* purpose. A data schema for purpose *payment* shown in Figure 3 is created automatically by system according to the privacy policies. The bilateral link is used to maintain the relationship between the source privacy object and its deputy privacy object. The data stored in the attributes, except attribute *OID*, of deputy classes (*~Know*, *Know*, *Consented*, *and Public*) is a pointer, which points to the corresponding attribute in the initial class (*patient*). The values of these attributes are computed by some switching operations when a data user tries to access the attributes.



patient

OID	*Name*	*Age*	*Address*	*CreditCard*	*Cost*
104224	Alice Park	10	123 First St Seattle WA	75852646112	2000
104228	Smith Parker	26	491 3rd St Lafayette IN	65745374531	3000
104324	Tom Hanks	32	35 Oval Dr Chicago IL	96543469931	6000
104224	Kerry Jones	34	541 6th St Lafayette WA	63548525144	1000

~Know

OID	*Address*
104265	----------------------
104276	---------------------
104336	---------------------
104290	---------------------

Know

OID	*Name*	*Age*	*CreditCard*	*Cost*
204224	----	--	-----------	----
434228	----	--	-----------	----
784324	----	--	-----------	----
504224	----	--	-----------	----

Consented

OID	*Age*
464224	--
484228	--
785624	--
554224	--

Public

OID	*Name*	*CreditCard*	*Cost*
464554	----	-----------	----
484268	----	-----------	----
785724	----	-----------	----
534224	----	-----------	----

Fig. 3. The class hierarchy schema for purpose payment

We suppose that for the purpose *payment*, a data provider does not have any privacy concern about privacy of the cost information when it is used by an accountant. Thus, the cost information can be used by an Accountant without any modification. And the data provider has great concerns about privacy of the cost information when it is used by a privacy database administrator; thus, the cost information should be used only in a sufficiently generalized form. Similarly, he has concerns about name information and does not have any privacy concern about credit-card information for *Accountant*, and has great concerns about name information and credit-card information for *Info_Admin*. The corresponding *SW* set is shown in Table 2. We suppose that for the purpose *payment*, the data provider assign the switch operation *sw1* for the role *Accountant*, and assign the switch operation *sw2* for thr role *Info_Admin*, respectively.

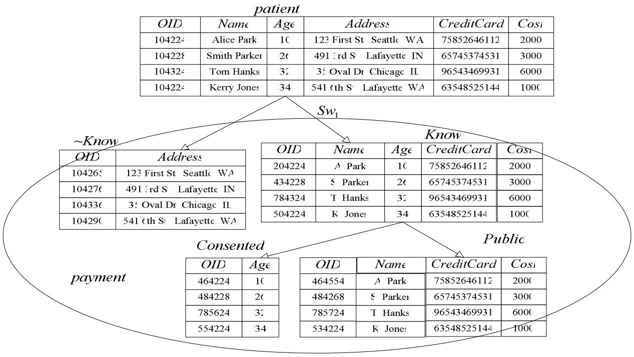
Note that, the *sw2* change the type of *Cost* attribute. The operation of changing data type can be supported in our model. So the data provider can select a proper data type to generalize his privacy data.

The switching operation Sw_1 is executed when a user with role *Accountant* accesses the privacy data class *potient* for purpose *payment*; the switching operation Sw_2 is executed when a user with role *Info_Admin* accesses the privacy data class *potient* for purpose *payment*. The results that can be obtained by role *Accountant* and role *Info_Admin* are shown in Figure 4 and Figure 5, respectively.

Supposed that the following query is executed for *payment* purpose: "SELECT * FROM *patient* WHERE OID = 104224". The system will return an object <'A.Parker', '75852646112', 2000> if the query is executed by a user with role *Accountant* and return an object <'A. P.', '7585264****', 1K-3K> if the query is executed by a user with role *Info_Admin*. So, different sets of data will be returned for the same query and the same purpose, depending on the role of user.

Table 2. The SWs defined for purpose payment

```
SwitchOperation sw1
{
  Patient.Name=GetFirstLetter(Patient.Name.FirstName)+
             "." + Patient.Name.LastName;
  Patient.Cost = Patient.Cost
 Patient. CreditCard= Patient. CreditCard
}
SwitchOperation sw2
{
 Patient.Name=     GetFirstLetter(Patient.Name.FirstName)
       +"."+GetFirstLetter (Patient.Name.LastName)
 If (0<=Patient.Cost<=3000)
       Patient.Cost = "1K-3K"
 elseif (3000<Patient.Cost<=5000)
       Patient.Cost = "3K-5K"
 elseif (5000<Patient.Cost<=7000)
       Patient.Cost = "5K-7K"
 Patient.CreditCard=GetSevenLetter(Patient.CreditCard)  +
                  "****"
}
```



OID	Name	Age	Address	CreditCard	Cost
104224	Alice Park	10	123 First St Seattle WA	75852646112	2000
104228	Smith Parker	26	491 3rd St Lafayette IN	65745374531	3000
104324	Tom Hanks	32	35 Oval Dr Chicago IL	96543469931	6000
104224	Kerry Jones	34	541 6th St Lafayette WA	63548525144	1000

OID	Address
104265	123 First St Seattle WA
104276	491 3rd St Lafayette IN
104336	35 Oval Dr Chicago IL
104290	541 6th St Lafayette WA

OID	Name	Age	CreditCard	Cost
204224	A Park	10	75852646112	2000
434228	S Parker	26	65745374531	3000
784324	T Hanks	32	96543469931	6000
504224	K Jones	34	63548525144	1000

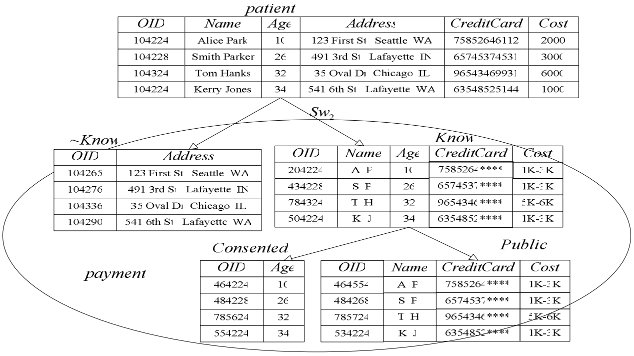
OID	Age
464224	10
484228	26
785624	32
554224	34

OID	Name	CreditCard	Cost
464554	A Park	75852646112	2000
484268	S Parker	65745374531	3000
785724	T Hanks	96543469931	6000
534224	K Jones	63548525144	1000

Fig. 4. The data obtained by role Accountant for purpose payment

4.3 Composite Privacy Object Model

A composite privacy object can be created by combining two or more atomic privacy objects according to the relationship among these privacy objects. The composite privacy object with common attributes and methods is grouped into a composite class. In our model, a composite privacy object can be expressed by a Join or a Union deputy object. If the combination operation is an aggregation of several privacy objects belonging to different classes respectively, a Join deputy object is selected. If the combination operation is a generalization of several privacy objects belonging to different classes respectively, a Union deputy object is selected. For example, in the



OID	Name	Age	Address	CreditCard	Cost
104224	Alice Park	1(	123 First St Seattle WA	75852646112	2000
104228	Smith Parker	26	491 3rd St Lafayette IN	6574537453l	300(
104324	Tom Hanks	32	35 Oval Dı Chicago IL	9654346993l	600(
104224	Kerry Jones	34	541 6th St Lafayette WA	63548525144	100(

OID	Address
104265	123 First St Seattle WA
104276	491 3rd St Lafayette IN
104336	35 Oval Dı Chicago IL
104290	541 6th St Lafayette WA

OID	Name	Age	CreditCard	Cost
20422-	A F	1(	758526-****	1K-3K
43422ε	S F	26	6574537****	1K-3K
784324	T H	32	965434(****	5K-6K
504224	K J	34	635485:****	1K-3K

OID	Age
464224	1(
48422ε	26
785624	32
554224	34

OID	Name	CreditCard	Cost
46455-	A F	758526-****	1K-3K
48426ε	S F	6574537****	1K-3K
785724	T H	965434(****	5K-6K
534224	K J	635485:****	1K-3K

Fig. 5. The data obtained by role Info_Admin for purpose payment

hospital system, if we want to express, *John is the doctor of Tom*, a Join deputy object is selected. And if we want to express, *Tom and Smith are living in breathing medicine*, a Union deputy object is to be selected. Once a composite object is created, it will be treated as an atomic privacy object, which can be further manipulated by the mechanism shown in section 4.2. The formal definition of composite privacy class is shown in Definition 3 and the creating process and data model is shown in Figure 6.

Definition 3. A composite privacy class can be defined as follows:

(1) A aggregation type of composite privacy class is defined as follows:

Let $PC_1 = \langle O_1^s, A_1^s, M_1^s \rangle$, $PC_2 = \langle O_2^s, A_2^s, M_2^s \rangle$, ... , $PC_k = \langle O_k^s, A_k^s, M_k^s \rangle$ denote source privacy classes, $ACPC = \langle O^d, A^d, M^d \rangle$ denotes an aggregation type of composite privacy class, then:

$$O^d = \{o^d \mid o^d \to (o_1^s \times o_2^s \times \ldots \times o_k^s) \in (O_1^s \times O_2^s \times \ldots \times O_k^s) \wedge SC(o_1^s \times o_2^s \times \ldots \times o_k^s) == \text{TRUE}\},$$

where $SC(o_1^s \times o_2^s \times \ldots \times o_k^s)$ is the constraint that combination operations must be satisfied.

(2) A generalization type of composite privacy class is defined as follows:

Let $PC_1 = \langle O_1^s, A_1^s, M_1^s \rangle$, $PC_2 = \langle O_2^s, A_2^s, M_2^s \rangle$, ... , $PC_k = \langle O_k^s, A_k^s, M_k^s \rangle$ denote source privacy classes, $GCPC = \langle O^d, A^d, M^d \rangle$ denotes a generalization type of composite privacy class, then:

$$O^d = \{o^d \mid o^d \to o^s, (o^s \in O_1^s \wedge SC_1(o^s) == \text{TRUE}) \vee \cdots \vee (o^s \in O_k^s \wedge SC_k(o^s) == \text{TRUE})\},$$

where $SC_i(o^s)$ is the constraint that generalization operations must be satisfied.

The following example illuminates how to generate a composite privacy object. In the hospital system, we add a *diagnosis* purpose, which needs to aggregate the privacy objects stored in the *patient* class and the *doctor* class, respectively. For *diagnosis* purpose, five privacy policies are added in the *privacy-policy* class, as shown in Table 3. The added policies in table 3 allow the *doctor* role to access the *Name* and *Age* attributes of *patient* class and prohibit anyone to access the *CreditCard* and *Cost* attribute of *patient* class for *diagnosis* purpose. The added policies in table 3 also prohibit anyone to access the *CreditCard* of *doctor* class and allow anyone to access

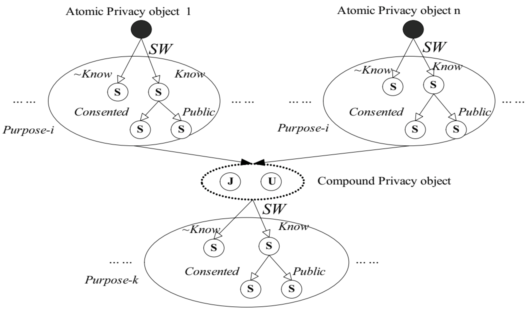


Fig. 6. Composite privacy object model

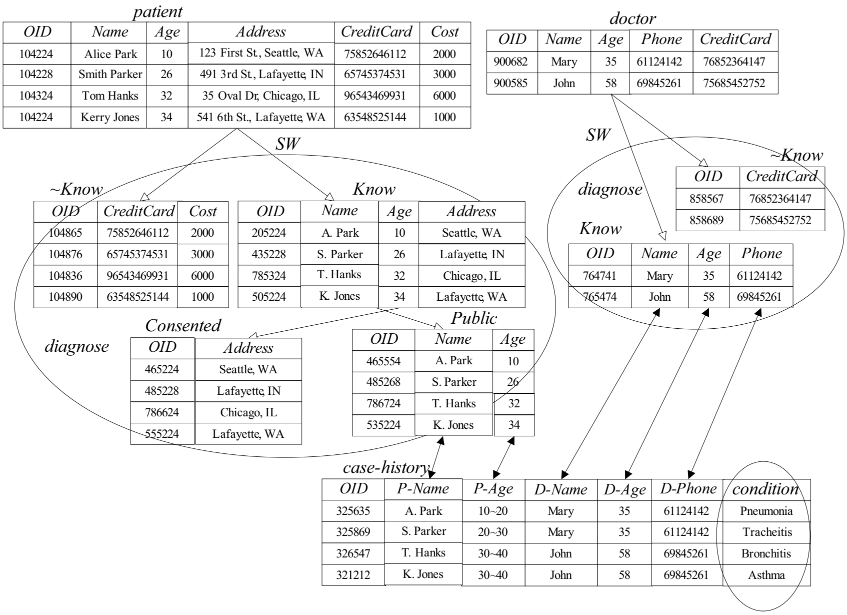
the other attributes of *doctor* class for *diagnosis* purpose. Though the positive policies for these attributes are not defined, we assume that if no positive privacy policy is defined for a privacy class, then all attributes, except those restricted by negative privacy policy, will be restricted by a default positive policy in our model.

Table 3. An extended privacy policy class

Purpose	Purpose Type	Class Name	Attribute Name	Externalrecipient	retention
payment	positive	patient	Name	{Accountant }	2007-1-7
payment	positive	patient	CreditCard	{Accountant }	2007-1-7
payment	positive	patient	Cost	{Accountant }	2007-1-7
payment	negative	patient	Address		
diagnosis	negative	doctor	CreditCard		
diagnosis	negative	patient	CreditCard		
diagnosis	negative	patient	Cost		
diagnosis	positive	patient	Name	{Doctor}	2007-3-7
diagnosis	positive	patient	Age	{Doctor}	2007-3-7

According to the assumption, all attributes of *doctor* class, except *CreditCard* attribute, are restricted by a default positive policy. So these attributes will be contained in *Public* deputy class, and *Know* deputy class is same as *Public* deputy class. In order to avoid data redundancy, only *Know* deputy class exists. As shown in Figure 7, for *diagnosis* purpose, only two deputy classes of *doctor* class, *~Know* and *Know*, are created. The operation result of creating composite privacy object for *diagnosis* purpose is shown in Figure 7. Here the corresponding definitions of *SW* and role for *diagnosis* purpose, which are similar to that for *payment* purpose, are omitted.

As shown in Figure 7, the result of completing *diagnosis* purpose will generate a new composite privacy class, *case-history* class. Each instance of the class is privacy information both doctor and patient. In *case-history* class, most attributes inherit from the corresponding source classes by deputy algebra. Some new privacy attributes, such as *condition* attribute, can be added to the privacy system. The example illustrates that it is easy for our model to support the extension of privacy data schema.



patient

OID	Name	Age	Address	CreditCard	Cost
104224	Alice Park	10	123 First St., Seattle, WA	75852646112	2000
104228	Smith Parker	26	491 3rd St., Lafayette, IN	65745374531	3000
104324	Tom Hanks	32	35 Oval Dr, Chicago, IL	96543469931	6000
104224	Kerry Jones	34	541 6th St., Lafayette, WA	63548525144	1000

doctor

OID	Name	Age	Phone	CreditCard
900682	Mary	35	61124142	76852364147
900585	John	58	69845261	75685452752

~Know

OID	CreditCard	Cost
104865	75852646112	2000
104876	65745374531	3000
104836	96543469931	6000
104890	63548525144	1000

Know

OID	Name	Age	Address
205224	A. Park	10	Seattle, WA
435228	S. Parker	26	Lafayette, IN
785324	T. Hanks	32	Chicago, IL
505224	K. Jones	34	Lafayette, WA

Consented

OID	Address
465224	Seattle, WA
485228	Lafayette, IN
786624	Chicago, IL
555224	Lafayette, WA

Public

OID	Name	Age
465554	A. Park	10
485268	S. Parker	26
786724	T. Hanks	32
535224	K. Jones	34

~Know

OID	CreditCard
858567	76852364147
858689	75685452752

Know

OID	Name	Age	Phone
764741	Mary	35	61124142
765474	John	58	69845261

case-history

OID	P-Name	P-Age	D-Name	D-Age	D-Phone	condition
325635	A. Park	10~20	Mary	35	61124142	Pneumonia
325869	S. Parker	20~30	Mary	35	61124142	Tracheitis
326547	T. Hanks	30~40	John	58	69845261	Bronchitis
321212	K. Jones	30~40	John	58	69845261	Asthma

Fig. 7. An example of composite privacy object model

Once a composite privacy object is created, we can treat it as an atomic privacy object in succeeding application. By executing the process recursively, we can constitute a larger, more complex privacy information system, such as a hospital or a large company privacy information database.

5 Implementation

Based on the privacy model discussed in above, we implemented a prototype of privacy database, called PD_TOTEM, based on the object deputy database, TOTEM. The TOTEM is developed by modifying the kernel of the PostgreSQL. In the PD_TOTEM, we extend the traditional SQL syntax with the definition syntax of the purpose and the assignment syntax of the privacy policies, and we add a release control module for preventing the released data disclosure.

In this section, we mainly discuss how to implement the atomic privacy model and how to execute the query process in PD_TOTEM. As discussed in above, the atomic privacy model is the basic of PD_TOTEM. The query operations are performed more frequently than the update operation and the insert operations, and the proposed privacy data model has a strong impact on the query operations.

When a data provider submits his privacy data, he will define a privacy policy set for the submitted data for each purpose. According to the privacy policy set, a data schema of the atomic privacy model for a special purpose will be derived and a default switching operation is defined. The default switching operation is executed only when the data provider access the privacy data. It is stored in a system schema, called *pg_switching*, which has been implemented in the TOTEM.

Subsequently, the data provider can assign which user/role can access his privacy and define a switching operation for a user/role. These switching operations constitute a switching operation set (SOS). In order to store the switching operations, a system schema, called *pg_userswitching*, is defined in PD_TOTEM. The system schema is defined as follows:

```
BasicClass pg_userswitching :
:Attribute
(switchname: NameData) //The Name of the switching operation
(purpose: NameData)   //The purpose for which the switching operation is called
(classid : Oid)       // Oid of the Class that the switching operation belongs to
(attrnum : int2)      // The No. of the attribute on which set this switching operation
(exprnum : int2)       // The No. of this switching operation defined in the attribute
                           with attrnum (Maybe there are several source attrs□so
                           need several switching operations for this the attribute)
(switching : text)     //The content of the switching operation
(rolid[1] : Oid)           //The roles that can use the switching operation
```

The rolid[1] is an array, each element of which is a user ID or a role ID that can call the switching operation. So, when a query is executed, the system knows which switching operation will be called for the role. The reason for creating the *pg_userswitching* class is to make sure there is one switching operation for each deputy class. In ODM, if no switching operation is defined on an attribute, the attribute can not be inherited by deputy class. The implementation framework of the atomic privacy model is illustrated in Figure 8.

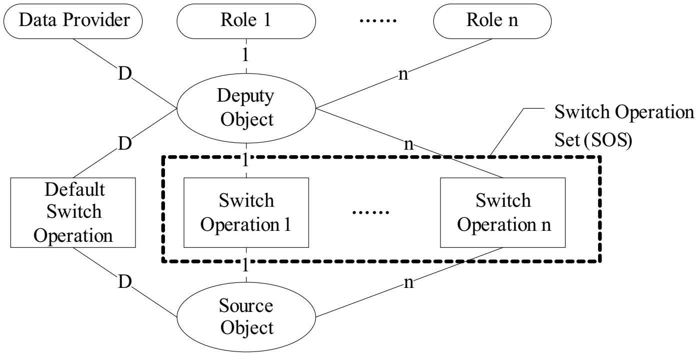


Fig. 8. The implementation framework of the atomic privacy model

When a data user u_i wants to access the privacy data with the purpose, the database system firstly authenticates the identity of the user. If the user is the data provider, he can access his privacy data along the path with "D" label. Otherwise, the system will select a proper switching operation sw_i in SOS for u_i according to the role of u_i and corresponding privacy policies. The data user u_i can only access the privacy data by the switching operation sw_i. As shown in Figure 8, the data user u_i can access his privacy data along the path with "i" label. So the different user/role can retrieve the different set of privacy data because the assigned switching operation is different.

6 Comparisons

In this section, we compare our privacy data model with the "NAIVE" model by the following criteria according to the example described in section 4.3. We assume that there are 1000 patient records and 50 doctor records in the example.

(1) Metadata storage

In the "NAIVE" model, the collected data is generalized and stored into multiple privacy levels. Supposed that there are *n* privacy levels in a privacy database system, the required storage space would be *n* times the size of the collected data in theory [2]. For example, in the example, the storage space required in "NAIVE" model can be computed by following equation:

$$1000 \times (m \times \sum_{i=1}^{6} \mathit{sizeof}(p\_a_i) + \mathit{sizeof}(\mathit{condition})) + 50 \times n \times \sum_{i=1}^{5} \mathit{sizeof}(d\_a_i)$$

Where *m* and *n* are privacy levels of "patient" class and "doctor" class, respectively. The *sizeof(p_a$_i$)* represents the allocated storage space for the attribute *p_a$_i$* contained in the "patient" class; *sizeof(d_a$_i$)* represent the allocated storage space for the attribute *d_a$_i$* contained in the "doctor" class, and *sizeof(condition)* represent the allocated storage space for the attribute *condition*.

In our model, each deputy attribute in deputy object only stores a pointer, which points to the corresponding attribute in source object. So, in the example, the storage space required in our model can be computed by following equation:

$$1000 \times (\sum_{i=1}^{6} \mathit{sizeof}(p\_a_i) + 4 \times \sum_{i=1}^{6} \mathit{sizeof}(\mathit{pointer}) + \mathit{sizeof}(\mathit{condition})) + 50 \times (\sum_{i=1}^{5} \mathit{sizeof}(d\_a_i) + 4 \times \sum_{i=1}^{5} \mathit{sizeof}(\mathit{pointer}))$$

Where *sizeof(pointer)* represents the allocated storage space for a pointer. In PD_TOTEM, a pointer is allocated 4 bytes. Apparently, the required storage space in our model is far less than that in the "NAIVE" model. The greater the required storage space of each attribute, the more obvious the advantage of our model.

(2) Query performance

The difference of query performance for atomic privacy data between our model and "NAIVE" model is not obvious. But for composite privacy data, the difference of query performance between the two models is very obvious.

In "NAIVE" model, composite privacy data is represented by a view, which is created by a join operation. If a view is created by a join table A, which has *m* records, and table B, which has *n* records, then the time complexity of querying a record in the view is *O(mn)*.

In our model, the join operation will create a Join deputy class. The time complexity of querying an object in the Join deputy class can be computed as follows. First, the query needs to traverse all objects in the Join deputy class. Then for each object, its source object will be directly found by the bilateral link (see [8, 9]). If there are *l* hops from the deputy object to its source object, the time complexity of querying a object in the Join deputy class is *O(ml)* or *O(nl)*. Because *l* is far smaller than *m* or *n*, the time complexity of our model is far smaller than that of the "NAIVE" model.

The reason that our model is more quickly than the "NAIVE" model for composite privacy data is that the cost of join operation is avoided in our model. When the Join deputy class is created, the structure of the object in the deputy class will be created. It is similar to the materialized view in the relational database. However, each cell of the Join deputy class stores a pointer rather than the corresponding data. It is similar to the virtual view in the relational database. It is the property that makes our model has the advantage of storage and query.

(3) Flexibility and scalability of data generation
In the "NAIVE" model, data generation is incorporated into the micro-views. If a data provider wants to update a method of data generation, he has two options: (1) delete the current view and regenerate a new view with same name; (2) update the definition of the current view. Moreover, the capability of generating the privacy data is only some simple operations, such as aggregation.

In our model, the switching operation is used to generalize privacy data. If a data provider wants to update a method of data generation, he can insert or change a switching operation object in the *pg_userswitching* class. Moreover, the switching operation can support complex generalization method, so the capability of hiding privacy data in our model is stronger than that in the previous model.

7 Conclusion and Future Work

In this paper, we proposed a privacy data model based on deputy mechanism. Furthermore, we discuss the implementation of the privacy data model. Compared with privacy data models based on view mechanism, our model has advantages on storage space and scalability of data generation. Comparing with conventional object view mechanism, our model pays more attention to the storage and usage of privacy data objects in the context of database.

Future works will include developing policy detection mechanism, such as how to detect privacy violations and how to detect the conflict between privacy policies in PD_TOTEM. In the current version of PD_TOTEM, we provide an interface that is used to define the switching operation by data provider. In future version of PD_TOTEM, the system will provide a GUI tool, by which a data provider can select the privacy level intuitively and the system will create the context of a switching operation automatically. And an access control mechanism based on deputy mechanism will be further explored.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic Databases. In: Proc. 28th Int'l Conf. on Very Large Data Bases, Hong Kong, China (2002)
2. Byun, J.-W., Bertino, E., Li, N.: Micro-views, or on how to protect privacy while enhancing data usability – Concepts and Challenges. SIGMOD Record 35(1) (2006)

3. Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5), 557–570 (2002)
4. Yao, C., Wang, L., Wang, X.S., Jajodia, S.: Indistinguishability: the Other Aspect of Privacy. In: Third International Workshop On Secure Data Management, Seoul, Korea (2006)
5. Xiao, X., Tao, Y.: Personalized Privacy Preservation. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, Chicago, IL, USA (2006)
6. Byun, J.-W., Bertino, E., Li, N.: Purpose Based Access Control of Complex Data for Privacy Protection. In: SACMAT 2005 (2005)
7. Byun, J.-W., Sohn, Y., Bertino, E., Li, N.: Secure Anonymization for Incremental Datasets. In: Third International Workshop On Secure Data Management, Seoul, Korea (2006)
8. Kambayashi, Y., Peng, Z.: An object deputy model for realization of flexible and powerful objectbases. Journal of Systems Integration, 329–362 (1996)
9. Peng, Z., Kambayashi, Y.: Deputy Mechanisms for Object-Oriented Databases. In: Proc. of IEEE 11th Int. Conf. on Data Engineering, pp. 333–340 (1995)
10. He, Q., Antón, A.I.: A Framework for Modeling Privacy Requirements in Role Engineering. In: International Workshop on Requirements Engineering for Software Quality (REFSQ 2003), Austria (2003)
11. Bertino, E., Sandhu, R.: Database Security-—Concepts, Approached, and Challenge. IEEE Transaction On Dependable and Secure Computing 2(1) (2005)
12. LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., DeWittResearch, D.: Limiting Disclosure in Hippocratic Databases. In: Proceedings of the 30th VLDB Conference, Toronto, Canada (2004)
13. Adam, N., Wortmann, J.: Security-control methods for statistical databases: A comparative study. ACM Computing Surveys (1989)
14. Garfinkel, S., Spafford, G.: Web Security,Privacy,and Commerce, 2nd edn. O'Reilly Media, Inc (2002)

Nominative Signature from Ring Signature

Dennis Y.W. Liu[1], Shuang Chang[1], Duncan S. Wong[1,⋆], and Yi Mu[2]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
{dliu,schang,duncan}@cs.cityu.edu.hk

[2] Centre for Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
ymu@uow.edu.au

Abstract. Since the introduction of nominative signature (NS) in 1996, there have been a handful of schemes proposed and almost all of them have been found flawed. The only one which is secure requires multi-round of communications between the nominator and the nominee for signature generation. In this paper, we propose a novel construction which is efficient and requires only one-move communication for signature generation. We also show that the construction is secure under the strongest security model currently available and the reductionist proofs only rely on standard number-theoretic assumptions. As of independent interest, our construction illustrates an interesting use of ring signature.

1 Introduction

A nominative signature (NS) involves three parties: *nominator* A, *nominee* B and *verifier* C. The nominator A arbitrarily chooses a message m and works jointly with the nominee B to produce a signature σ called nominative signature. The validity of σ can only be verified by B and if σ is valid, B can convince the verifier C the validity of σ using a *confirmation protocol*; otherwise, B can convince C the invalidity of σ using a *disavowal protocol*. Below are the properties of a nominative signature summarized from [11,9,18,8,13].

1. (*Joint Work of Nominator and Nominee*) A or B alone is not able to produce a valid σ;
2. (*Only Nominee Can Determine the Validity of Signature*) Only B can verify σ;
3. (*Can Only be Verified with Nominee's Consent*) The validity of σ is only verifiable with the aid of B, by running a confirmation/disavowal protocol with B;

[⋆] The author was supported by CityU grants (Project Nos. 7001959 and 7002001).

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 396–411, 2007.

© Springer-Verlag Berlin Heidelberg 2007

4. (*Nominee Cannot Repudiate*) If σ is valid, B cannot mislead C to believe that σ is invalid using the disavowal protocol. If σ is invalid, B cannot mislead C to believe that σ is valid using the confirmation protocol;
5. (*Nominator Chooses Message*) Message m is chosen by A;
6. (*Only Nominator Can Nominate*) B is chosen/nominated by A.

Since the introduction of nominative signature (NS) [11], it has been considered as a dual scheme of undeniable signature (US) [4,5]. Both nominative signature and undeniable signature are *non-self-authenticating*, namely, the public is not able to determine if a signature-message pair is valid or not merely from the signature itself and the signer's public key. For US, it can only be verified with the aid of the signer, while for NS, it can only be verified with the aid of the nominee, rather than the nominator (albeit it is the nominator who chooses the message). Nominative signature is also related to designated verifier signature (DVS) [10], designated confirmer signature (DCS) [3] and universal designated-verifier signature (UDVS) [17]. Due to the page limitation, we refer readers to [13] for a detailed comparison among these signature types. We emphasize that among all these signature types, only NS has the privilege of proving the signature validity been dethroned from the signer who chooses the message. None of the other signature types has this property.

Due to the special property of NS that the nominator cannot convince anyone about the validity of a nominative signature while only the nominee can, NS has been found [13] to be very useful for implementing *user certification systems*. A user certification system allows a user B to show the validity of a certificate issued by an authority A. The certificate can be B's birth certificate, his driving licence or one of his academic transcripts. Without using nominative signature, say using UDVS instead[1], B has to trust A not to validate B's certificates to anyone else without B's consent. This is mainly due to the privacy concern of B. If nominative signature is used, B does not need to trust A as A cannot validate B's certificate to anyone. Only B can do the proof. In addition, nominative signature ensures that B cannot forge such a certificate without A's consent. For more details and other applications, we refer readers to [13].

Related Work. The notion and construction of nominative signature were first proposed by Kim, Park and Won [11]. However, their construction was later found flawed by Huang and Wang [9]. In the construction of [11], the nominator can always determine the validity of a nominative signature, that is, violating Property 2 of nominative signature. In [9], the notion of convertible nominative signature was proposed, aiming at allowing only the nominee to convert a signature to a publicly-verifiable one. They also proposed a new scheme. However, it was later found [18,8] that the nominator in their scheme can generate valid signatures on his own and show the validity of the signature to anyone without the help of the nominee. That is, their scheme does not satisfy Properties 1 to 3. In [9], a definition and some requirements for nominative signature

[1] Using UDVS to implement a user certification system was first proposed by Steinfeld et al. in [17].

were proposed. However, their definition of nominative signature does not match with the scheme they proposed and the set of security requirements specified are incomplete and informal.

In [13], Liu et al. proposed the first formal security model for NS and also a proven secure construction. Their construction requires multi-round communications between the nominator and the nominee for signature generation. It is currently unknown if a one-move NS scheme can be built. In this paper, we answer this question positively by proposing a proven secure NS scheme which requires only one-move communication.

Our Results. We propose a novel construction of nominative signature. Comparing with the existing one [13], which is proven secure under the strongest security model currently available, our scheme requires only one-move communication between the nominator and the nominee for signature generation, while the existing one requires multi-round of communications. We also show that the reductionist proofs for the security of our scheme only rely on some standard number-theoretic assumptions.

Paper Organization. The definition of nominative signature and corresponding adversarial models are specified and discussed in Sec. 2. Our nominative signature scheme is described in Sec. 3. Its security is then analyzed in Sec. 4. Finally, the paper is concluded in Sec. 5.

2 Definition and Security Model

A nominative signature (NS) consists of three algorithms ($\mathsf{SystemSetup}$, KeyGen, $\mathsf{Ver}^{\mathsf{nominee}}$) and three protocols ($\mathsf{SigGen}$, $\mathsf{Confirmation}$, $\mathsf{Disavowal}$).

1. $\mathsf{SystemSetup}$ (System Setup): On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a list of system parameters denoted by param.
2. KeyGen (User Key Generation): On input param, it generates a public/private key pair (pk, sk).
3. $\mathsf{Ver}^{\mathsf{nominee}}$ (Nominee-only Verification): On input a message m, a nominative signature σ, a public key pk_A and a private key sk_B, it returns valid or $\mathsf{invalid}$.

An NS proceeds as follows. Given a security parameter $k \in \mathbb{N}$, $\mathsf{SystemSetup}$ is invoked and param is generated. KeyGen is then executed to initialize each party that is to be involved in the subsequent part of the scheme. One party called nominator is denoted by A. Let (pk_A, sk_A) be the public/private key pair of A. Let B be the nominee that A nominates, and (pk_B, sk_B) be B's public/private key pair. In the rest of the paper, we assume that entities can be uniquely identified from their public keys. To generate a nominative signature σ, A chooses a message $m \in \{0,1\}^*$, and carries out SigGen protocol with B. The protocol is defined as follows.

SigGen Protocol: Common inputs of A and B are param and m. A's additional input is pk_B, indicating that A nominates B as the nominee; and B's additional input is pk_A indicating that A is the nominator. At the end, either A

or B outputs σ. The party who outputs σ should be explicitly indicated in the actual scheme specification.

Signature Space: A value σ is a nominative signature with respect to pk_A and pk_B if it is in the *signature space* of the NS with respect to pk_A and pk_B. We emphasize that the signature space has to be specified explicitly in each actual NS scheme.

The validity of a nominative signature σ on message m (with respect to pk_A and pk_B) can be determined by B as $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$. To convince a third party C on the validity or invalidity of (m, σ, pk_A, pk_B), B as a prover and C as a verifier carry out the Confirmation or Disavowal protocol as follows.

Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B), B sets μ to 1 if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$; otherwise, μ is set to 0. B first sends μ to C. If $\mu = 1$, Confirmation protocol is carried out; otherwise, Disavowal protocol is carried out. At the end of the protocol, C outputs either accept or reject while B has no output.

Correctness. Suppose that all the algorithms and protocols of a nominative signature scheme are carried out accordingly by honest entities A, B and C, the scheme satisfies the correctness requirement if

1. $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$; and
2. C outputs accept at the end of the Confirmation protocol.

Validity of a Nominative Signature. A nominative signature σ on message m with respect to nominator A and nominee B is *valid* if $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B) = \mathsf{valid}$. In this case, we say that quadruple (m, σ, pk_A, pk_B) is *valid*. Note that only B can determine the validity of σ (Property 2).

In the following, we review the four formal security models defined in [13]. They capture the security notions of (1) unforgeability, (2) invisibility, (3) security against impersonation, and (4) non-repudiation.

2.1 Unforgeability

The following definition captures Property 1, 5 and 6 (Sec. 1).

Game Unforgeability: Let $\mathcal{S}$ be the simulator and $\mathcal{F}$ a forger. Let $k \in \mathbb{N}$ be a security parameter.

1. (*Initialization*) First, $\mathsf{param} \leftarrow \mathsf{SystemSetup}(1^k)$ is executed and key pairs (pk_A, sk_A) and (pk_B, sk_B) for nominator A and nominee B, respectively, are generated using KeyGen. Then $\mathcal{F}$ is invoked on input $(\mathsf{param}, pk_A, pk_B)$.
2. (*Attacking Phase*) $\mathcal{F}$ can make queries to the following oracles:
   - CreateUser: On input an identity I, execute $(pk_I, sk_I) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, and output pk_I.
   - Corrupt: On input a public key pk, if pk is generated by CreateUser or in $\{pk_A, pk_B\}$, the corresponding private key is returned; otherwise, $\perp$ is returned. pk is said to be *corrupted*.

- SignTranscript: On input a message m, two distinct public keys, pk_1 (the nominator) and pk_2 (the nominee) such that at least one of them is uncorrupted, and one parameter called $role \in \{\mathsf{nil}, \mathsf{nominator}, \mathsf{nominee}\}$: (1) if $role = \mathsf{nil}$, $\mathcal{S}$ simulates a run of SigGen and returns a valid quadruple (m, σ, pk_1, pk_2) and $trans_\sigma$ which is the transcript of the execution of SigGen; (2) if $role = \mathsf{nominator}$, $\mathcal{S}$ (as nominee with public key pk_2) simulates a run of SigGen with $\mathcal{F}$ (as nominator with pk_1); (3) if $role = \mathsf{nominee}$, $\mathcal{S}$ (as nominator with pk_1) simulates a run of SigGen with $\mathcal{F}$ (as nominee with public key pk_2).
- Confirmation/disavowal: On input a message m, σ and two public keys pk_1 (the nominator), pk_2 (the nominee), such that σ is in the signature space with respect to pk_1 and pk_2, let sk_2 be the corresponding private key of pk_2, the oracle responds based on whether a passive attack or an active/concurrent attack is mounted: (1) In a passive attack, if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2)$, the oracle returns a bit $\mu = 1$ and a transcript of the Confirmation protocol. Otherwise, $\mu = 0$ and a transcript of the Disavowal protocol are returned. (2) In an active/concurrent attack, if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2)$, it returns $\mu = 1$ and executes the Confirmation protocol with $\mathcal{F}$ (acting as a verifier). Otherwise, it returns $\mu = 0$ and executes the Disavowal protocol with $\mathcal{F}$. $\mathcal{F}$ interacts serially with the oracle in the active attack while $\mathcal{F}$ interacts with different instances of the oracle concurrently in the concurrent attack.

3. (*Output Phase*) $\mathcal{F}$ outputs a pair (m^*, σ^*) as a forgery of A's nominative signature on message m^* with B as the nominee.

$\mathcal{F}$ *wins* the game if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B)$ and (1) $\mathcal{F}$ does not corrupt both sk_A and sk_B; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to SignTranscript for any $role$; (3) $(m^*, \sigma', pk_A, pk_B)$ has never been queried to Confirmation/disavowal for any σ' in the signature space with respect to pk_A and pk_B (check *Signature Space* on page 399). $\mathcal{F}$'s advantage is defined to be the probability that $\mathcal{F}$ wins.

Definition 1 ([13]). *An NS scheme is unforgeable if no PPT forger $\mathcal{F}$ has a non-negligible advantage in Game Unforgeability.*

2.2 Invisibility

Game Invisibility: The initialization phase is the same as that of Game Unforgeability. Let $\mathcal{D}$ be a distinguisher (i.e. the adversary) that can query any of the oracles described in Game Unforgeability. At some point in the attacking phase, $\mathcal{D}$ outputs a message m^* and requests for a challenge NS σ^* on m^*. The challenge σ^* is generated based on the outcome of a hidden coin toss b. If $b = 1$, σ^* is generated by running SigGen. If $b = 0$, σ^* is chosen randomly from the signature space with respect to pk_A and pk_B. At the end of the game, $\mathcal{D}$ outputs a guess b'.

$\mathcal{D}$ *wins* if $b' = b$ and (1) $\mathcal{D}$ does not corrupt sk_B; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to SignTranscript; (3) $(m^*, \sigma^*, pk_A, pk_B)$ has never been queried to Confirmation/disavowal. $\mathcal{D}$'s advantage in this game is defined as $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 2 ([13]). *An NS scheme has the property of invisibility if no PPT distinguisher $\mathcal{D}$ has a non-negligible advantage in Game Invisibility.*

2.3 Security Against Impersonation

Game Impersonation: The initialization phase is the same as that of Game Unforgeability. The game has two additional phases. Let $\mathcal{I}$ be an impersonator (i.e. adversary).

- (*Preparation Phase*) In this phase, $\mathcal{I}$ may query any of the oracles defined in Game Unforgeability. $\mathcal{I}$ prepares (m^*, σ^*, μ) where m^* is some message, σ^* is in the signature space with respect to pk_A and pk_B and μ is a bit.
- (*Impersonation Phase*) If $\mu = 1$, $\mathcal{I}$ (as nominee) executes Confirmation protocol with the simulator (as a verifier). If $\mu = 0$, $\mathcal{I}$ executes Disavowal protocol instead.

$\mathcal{I}$ *wins* if the simulator outputs accept at the Impersonation Phase while $\mathcal{I}$ has never corrupted sk_B in the game. $\mathcal{I}$'s advantage is defined to be the probability that $\mathcal{I}$ wins.

Definition 3 ([13]). *An NS scheme is secure against impersonation if no PPT impersonator $\mathcal{I}$ has a non-negligible advantage in Game Impersonation.*

2.4 Non-repudiation

Game Non-repudiation: The initialization phase is the same as that of Game Unforgeability. Let $\mathcal{B}$ be a *cheating* nominee which can query any of the oracles defined in Game Unforgeability. The game also has two additional phases. (1) (*Preparation Phase*) $\mathcal{B}$ prepares (m^*, σ^*, μ) where m^* is a message and σ^* is in the signature space with respect to pk_A and pk_B. $\mu = 1$ if $\mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B) =$ valid; otherwise, $\mu = 0$. (2) (*Repudiation Phase*) If $\mu = 1$, $\mathcal{B}$ executes Disavowal protocol with the simulator (acting as a verifier) on $(m^*, \sigma^*, pk_A, pk_B)$ but the first bit sent to the simulator is 0. If $\mu = 0$, $\mathcal{B}$ executes Confirmation protocol but the first bit sent to the simulator is 1.

$\mathcal{B}$ *wins* the game if the simulator outputs accept in the repudiation phase. $\mathcal{B}$'s advantage is defined to be the probability that $\mathcal{B}$ wins.

Definition 4 ([13]). *An NS scheme is secure against repudiation by nominee if no PPT cheating nominee $\mathcal{B}$ has a non-negligible advantage in Game Non-repudiation.*

Remark: By the soundness property of a proof system, if the system is for a language L over $\{0,1\}^*$, then if $\lambda \notin L$, the verifier will accept with probability at most δ which is over $[0, \frac{1}{2})$.

3 Our Construction

In this section, we construct a nominative scheme using ring signature. A ring signature, introduced by Rivest, Shamir and Tauman [15], associates with a group of members in such a way that it allows any member within the group, called "a ring", to sign a message on behalf of the ring, without collaborating with any other ring members. Also, the signature does not reveal about who the actual signer is. For our NS construction, it is a special use of ring signature as we restrict ourselves to the case of having two signers in the ring only. Our technique is related to the property of *claimability* of ring signature. This property allows the actual signer of a ring signature to later prove its authorship of the signature [15]. This property has been used implicitly in [6] to construct a concurrent signature scheme.

Two-member Ring Signature. In Appendix A of Abe et al.'s Asiacrypt 2002 paper [1], there is an efficient ring signature scheme proposed. Below is a review for a two-member case.

Let G be a cyclic group of prime order q. Let g be a generator of G. Let (x_i, y_i) be the private/public key pair of user i, for $i = 1, 2$, $x_i \in_R \mathbb{Z}_q$ and $y_i = g^{x_i}$. Let $L = (y_1, y_2)$. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function. Without loss of generality, below is the signature generation carried out by user 1.

1. Randomly pick $c_2, r \in_R \mathbb{Z}_q$ and compute $z = g^r y_2^{c_2}$.
2. Compute

$$\begin{aligned} c &= H(L\|m\|z) \\ c_1 &= c - c_2 \bmod q \\ s &= r - c_1 x_1 \bmod q \end{aligned}$$

The ring signature is $\sigma = (s, c_1, c_2)$. (L, m, σ) is valid if

$$c_1 + c_2 \equiv H(L\|m\|g^s y_1^{c_1} y_2^{c_2}) \pmod{q}. \tag{1}$$

An NS Construction. (*Idea*) In the two-member ring signature above, we can see that for a given pair (c_2, r), the value of c_1 and other variables in the signature generation are determined. In addition, due to the random oracle assumption of H, c_1 behaves like a random instance in $\mathbb{Z}_q$. In our NS construction, we let the nominee B generate a specially formed c_2 first and then have the nominator A generate values for other variables of σ (i.e. s and c_1). The specially formed c_2 allows B to carry out confirmation/disavowal protocol while the rest of our construction ensures that A cannot carry out similar proofs due to the fact that it is unlikely for c_1 to possess the special form as that of c_2. Below are the details.

SystemSetup: Let $k \in \mathbb{N}$ be a security parameter. Randomly pick a prime $q' \geq 2^k$ such that both $q = 2q'+1$ and $p = 4q'+3$ are prime. Randomly pick an integer $g \in_R \mathbb{Z}_p^*$ such that the order of g mod p is q. Randomly pick another integer

$h \in_R \mathbb{Z}_q^*$ such that the order of h mod q is $q-1$. Let $H : \{0,1\}^* \to \mathbb{Z}_q^*$ be a hash function. The list of system parameters is $\mathsf{param} = (k, p, q, g, h, H)$[2].

KeyGen: On input param, it generates (x', x'', y', y'') where $x', x'' \in_R \mathbb{Z}_q^*$, $y' = g^{x'} \bmod p$ and $y'' = h^{x''} \bmod q$. Let $((x_i', x_i''), (y_i', y_i''))$ be the private/public key pair of entity i, for $i = A$ or B, corresponding to nominator and nominee, respectively. Let $L = ((y_A', y_A''), (y_B', y_B''))$.

SigGen Protocol: On common input param and some message $m \in \{0,1\}^*$, and specific input y_B' for A and y_A' for B, the protocol proceeds as follows.

1. B computes $c_2 = H(m\|L)^{x_B''} \bmod q$ and sends c_2 to A.
2. A carries out the remaining steps of the ring signature generation, that is, finding $s, c_1 \in \mathbb{Z}_q$ such that

$$c_1 \cdot c_2 \equiv H(L\|m\|g^s {y_A'}^{c_1} {y_B'}^{c_2} \bmod p) \pmod q \tag{2}$$

   A outputs a nominative signature $\sigma = (s, c_1, c_2)$, which is actually a ring signature[3].

Signature Space (page 399). We say that σ is a *nominative signature* if (L, m, σ) is a valid ring signature (i.e. satisfying Eq. (2)).

$\mathsf{Ver}^{\mathsf{nominee}}$: On input (L, m, σ) where $\sigma = (s, c_1, c_2)$ is a nominative signature (i.e. in the signature space defined above), if $c_2 = H(m\|L)^{x_B''} \bmod q$, output valid; otherwise, $\mathsf{invalid}$.

Confirmation/Disavowal Protocol: On input (L, m, σ) where $\sigma = (s, c_1, c_2)$ is a nominative signature (as defined above), B first checks if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, y_A', (x_B', x_B''))$. If so, B sends $\mu = 1$ to verifier C and proves to C that $(h, y_B'', H(m\|L), c_2)$ is a DH-tuple using a WI protocol. Otherwise, B sends $\mu = 0$ to C and shows that $(h, y_B'', H(m\|L), c_2)$ is a non-DH-tuple using a WI protocol.

We say that (h, h^a, h^b, h^c) is a DH-tuple in $\mathbb{Z}_q^*$ if $c \equiv ab \pmod{q-1}$; otherwise, it is a non-DH-tuple. According to [12], Witness Indistinguishable (WI) [7] protocols can be used to prove/disprove a DH-tuple, that is, it is sufficient for the prover to execute the protocols successfully using its knowledge of either one of the witnesses, i.e. a or b. In the Confirmation/Disavowal protocol above, B's knowledge is x_B''. For concrete implementation, we use the protocols described by Kurosawa and Heng [12].

[2] Note that $p = 2q + 1$. g generates a *subgroup* $\mathfrak{G} \subset \mathbb{Z}_p^*$ of order q and h generates $\mathbb{Z}_q^*$. Half of the elements in $\mathbb{Z}_p^*$ are generators of $\mathfrak{G}$ and half of the elements in $\mathbb{Z}_q^*$ are generators of $\mathbb{Z}_q^*$.

[3] To ensure that both c_1 and c_2 are in the multiplicative group $\mathbb{Z}_q^*$, rather than in the additive group $\mathbb{Z}_q$, we change the modular addition of c_1 and c_2 as in Eq. (1) to a modular multiplication in Eq. (2). This is for invisibility since the nominee always generates a value of c_2 in $\mathbb{Z}_q^*$ rather than in $\mathbb{Z}_q$.

Discussions. The specially formed $c_2 = H(m\|L)^{x''_B} \bmod q$ is actually an undeniable signature (due to Chaum [4,12]). Obviously, we should require that the discrete logarithm (DLOG) problem in $\mathbb{Z}_q^*$ is intractable, namely, given a pair of random instances $(h, c_2) \in \mathbb{Z}_q^*$, it is infeasible to find x such that $c_2 \equiv h^x \pmod q$. This is ensured in our scheme by requiring that $q = 2q' + 1$, that is, $q-1$ contains at least one large prime $q' \geq 2^k$.

In the definition of signature space above (i.e. a nominative signature), note that even σ is a valid ring signature on (L, m), that is, σ is a nominative signature, the validity of the nominative signature σ is undetermined. We can see that given L, it is easy to evaluate if σ falls in the signature space defined above (i.e. σ is a nominative signature but its validity can only be determined by B). Therefore, when evaluating $\mathsf{Ver}^{nominee}$, we can assume that σ has already been checked to be a nominative signature.

The SigGen Protocol requires only one-move message transfer from B to A. No multi-round interaction is required. Furthermore, the nominative signature does not contain any standard signature. Hence from the nominative signature itself, a third party is not able to identify any particular entity who must have involved during the signature generation stage. Note that a nominative signature σ, as defined, is only a valid ring signature on some message m with respect to L, it can be generated by nominator A or nominee B alone, and any third party is not able to find out exactly who the actual signer is.

In SystemSetup, we need to randomly choose a large prime q' such that both $2q' + 1$ and $4q' + 3$ are prime. The following proposition claims that there are adequately large number of such primes.

Conjecture 1. For any sufficiently large positive integer n, there are at least $\frac{n}{(\log n)^3}$ primes in $[1, n]$ such that for each of these primes, denoted by p, $2p + 1$ and $4p + 3$ are also prime.

Proof. We apply Shoup's Hypothesis H conjecture [16]. Below is a review of the conjecture.

Hypothesis H Conjecture. Let $(a_1, b_1), ..., (a_k, b_k)$ be distinct pairs of integers such that $a_i > 0$. Let $\pi(n)$ be the number of integers u up to n such that $ua_i + b_i$ are simultaneously prime for $i = 1, \cdots, k$. Then

$$\pi(n) \sim D \frac{n}{(\log n)^k}$$

where

$$D \doteq \prod_p (1 - \frac{1}{p})^{-k} (1 - \frac{\omega(p)}{p}),$$

with the product taken over all primes p, and $\omega(p)$ being the number of distinct solutions of $u \bmod p$ to the congruence

$$\prod_{i=1}^{k} (ua_i + b_i) \equiv 0 \pmod p. \tag{3}$$

By applying the Hypothesis H Conjecture above, we can estimate the density of the primes stated in the theorem. To do this, we set $k = 3$ and (a_1, b_1), (a_2, b_2), (a_3, b_3) to $(1, 0)$, $(2, 1)$, $(4, 3)$, respectively. According to (3), the number of integers u up to n such that u, $2u + 1$, $4u + 3$ are primes should be

$$\pi(n) \sim D\frac{n}{(\log n)^3}.$$

To estimate the value of D, we consider three case. For the case $p = 2$, $\omega(p) = 1$ as there is only one solution to the congruence of (3). Let

$$D_1 \doteq \prod_{p=2}(1-\frac{1}{p})^{-k}(1-\frac{\omega(p)}{p}) = (1-\frac{1}{2})^{-3}(1-\frac{1}{2}) = 4.$$

For the case $p = 3$, $\omega(p) = 2$ as there are two distinct solutions to the congruence of (4). Let

$$D_2 \doteq \prod_{p=3}(1-\frac{1}{p})^{-k}(1-\frac{\omega(p)}{p}) = (1-\frac{1}{3})^{-3}(1-\frac{2}{3}) = \frac{9}{8}.$$

For the case $p > 3$, there are two subcases. If p is in the form $4m + 3$ for some integer m, $\omega(p) = 3$. These three solutions of $u \bmod p$ to the congruence of (3) are when $u \bmod p$ is congruent to 0, m and $2m + 1$. Note that for all $p > 3$, since they are all odd primes, p must be in the form of $2m + 1$. Hence the other subcase is when $p > 3$ but it is not in the form $4m+3$. In this subcase, p can only be represented in the form of $2m + 1$. We have $\omega(p) = 2$. These two solutions are when $u \bmod p$ is congruent to 0 and m. Let

$$D_3 \doteq \prod_{p>3}\frac{p^2}{(p-1)^3}\prod_{p'\in P'}(p'-3)\prod_{p''\in P''}(p''-2).$$

where P' is the set of all primes in the form of $4m + 3$, where m is an integer, and P'' is the set of all primes greater than 3 but not in the form of $4m + 3$. Note that $P' \cup P''$ is the set of all primes greater than 3, and P' and P'' are disjoint.

By combining these three cases, we get

$$D \doteq D_1 D_2 D_3 = 4.5\prod_{p>3}\frac{p^2}{(p-1)^3}\prod_{p'\in P'}(p'-3)\prod_{p''\in P''}(p''-2) \approx 10.2153$$

for $n = 2^{20}$. Since $D = 4.5\prod_{p'\in P'}\frac{p'^2(p'-3)}{(p'-1)^3}\prod_{p''\in P''}\frac{p''^2(p''-2)}{(p''-1)^3} > 4.5 \cdot \frac{1}{2} \cdot \frac{1}{2} > 1$, we have the density of the primes stated in the theorem estimated as follows.

$$\frac{\pi(n)}{n} = \frac{D}{(\log n)^3} > \frac{1}{(\log n)^3}$$

□

For comparison, we can use the similar method to find out that there are at least $\frac{n}{(\log n)^2}$ Sophie Germain primes in $[1, n]$.

4 Security Analysis

It is obvious that the construction above satisfies the correctness requirement. In the following, we show that the construction also satisfies the four security requirement defined in Sec. 2.

Let (t, ϵ, Q)-$\mathcal{A}$ be a PPT algorithm $\mathcal{A}$ which solves a pre-specified problem with probability at least ϵ after running for at most time t and making at most Q queries in the underlying game. Similarly, (t', ϵ')-$\mathcal{B}$ denotes a PPT algorithm which runs for at most time t' and with success probability of at least ϵ'.

Lemma 1 (Cheating Nominee). *For the NS scheme proposed above, given nominee B's private key, if a (t, ϵ, Q)-forger wins* Game Unforgeability*, there exists a $(2t + 2Qt_o + c, (\epsilon/4)^2)$-adversary which can solve the DLOG problem in the subgroup of $\mathbb{Z}_p^*$ of order q generated by g, where t_o is the maximum time for simulating one oracle query and c is some constant.*

Proof. If a forger $\mathcal{F}$, after obtaining B's private key (x'_B, x''_B) via Corrupt, wins Game Unforgeability, we construct an algorithm $\mathcal{S}$ which solves a random DLOG problem instance $y^* = g^{x^*} \bmod p$.

Game Simulation: At the beginning of the simulation of Game Unforgeability, $\mathcal{S}$ generates param according to SystemSetup, and sets nominator A's public key (y'_A, y''_A) to $(y^*, h^{x''_A} \bmod q)$, where $x''_A \in_R \mathbb{Z}_q^*$. For nominee B, the public/private key pair is generated using KeyGen accordingly. A List is maintained for keeping track of the queries and answers of random oracle H.

When $\mathcal{F}$ is invoked, 1^k, $pk_A = (y'_A, y''_A)$ and $pk_B = (y'_B, y''_B)$ are given to $\mathcal{F}$ and oracles are simulated as follows. For a query to random oracle H, List is checked to see if the same query has made before, if so, the same answer will be returned; otherwise, $\mathcal{S}$ randomly picks $r \in_R \mathbb{Z}_{q-1}$ and sets the answer to $h^r \bmod q$. List is updated accordingly. For a query to CreateUser, a new public/private key pair is generated according to KeyGen and the public key is returned. If a Corrupt query is received, for example, when B is queried, (x'_B, x''_B) is returned. As restricted, $\mathcal{F}$ cannot query Corrupt for A's private key. For a SignTranscript query, there are three cases.

- Case (1): If $role =$ nil, a nominative signature is simulated by following SigGen. There is one exception: when A is indicated as a nominator or a nominee in query, $\mathcal{S}$ is not able to generate a valid nominative signature using SigGen. Thanks to random oracle, $\mathcal{S}$ can still generate a valid nominative signature with distribution indistinguishable from the one generated using SignGen by following the steps below. (1) Set $H(m \| pk_1 \| pk_2) = h^r \bmod q$ for a random $r \in_R \mathbb{Z}_{q-1}$ and update List with $(m \| pk_1 \| pk_2,\ h^r \bmod q, r)$; (2) Compute $c_2 = {y''_2}^r \bmod q$ where $pk_2 = (y'_2, y''_2)$; (3) randomly choose $c_1 \in_R \mathbb{Z}_q^*$ and $s \in_R \mathbb{Z}_q$; set $H(pk_1 \| pk_2 \| m \| g^s {y'_1}^{c_1} {y'_2}^{c_2} \bmod p)$ to $c_1 \cdot c_2 \bmod q$ and update List accordingly, where $pk_1 = (y'_1, y''_1)$; (4) return $(m, (s, c_1, c_2), pk_1, pk_2)$ and $trans_\sigma = c_2$. This simulation is computationally indistinguishable from a real simulation due to the idealness of random oracles and uniform distributions of s, c_1 and c_2 in their corresponding domains.

- Case (2): If *role* = nominator, $\mathcal{S}$, acts as the nominee, carries out the SigGen protocol with $\mathcal{F}$. Similar to Case (1) above, if A is the nominee, $\mathcal{S}$ simply follows steps (1) and (2) above to perform the role of the nominee.
- Case (3): If *role* = nominee, $\mathcal{S}$, acts as nominator, carries out the SigGen protocol with $\mathcal{F}$. Similar to Case (1) above, if A is the nominator, $\mathcal{S}$ simply follows steps (3) and (4) above to perform the role of the nominator.

For a Confirmation/disavowal query $(m, \sigma = (s, c_1, c_2), pk_1, pk_2)$, since $\mathcal{S}$ always knows the discrete logarithm of $H(m\|pk_1\|pk_2)$ to the base h modulo q, $\mathcal{S}$ can always carry out the confirmation/disavowal protocols with respect to DH-tuples/non-DH-tuples.

Reduction: For one successful simulation, suppose the forgery of $\mathcal{F}$ is $\sigma^1 = (s^1, c_1^1, c_2^1)$ on some message m^* such that

$$c_1^1 \cdot c_2^1 \equiv H(L\|m^*\|g^{s^1} {y'_A}^{c_1^1} {y'_B}^{c_2^1} \bmod p) \pmod q$$

and $c_2^1 = H(m^*\|pk_A\|pk_B)^{x''_B} \bmod q$, by the assumption of random oracle model, $\mathcal{F}$ has a query $H(L\|m^*\|g^{s^1} {y'_A}^{c_1^1} {y'_B}^{c_2^1} \bmod p)$. After a successful rewind simulation [14,1], $\mathcal{F}$ outputs another forgery $\sigma^2 = (s^2, c_1^2, c_2^2)$ on the same message m^*. We have

$$c_1^1 \cdot c_2^1 \not\equiv c_1^2 \cdot c_2^2 \pmod q \tag{4}$$

$$(c_1^1 - c_1^2)x^* + (c_2^1 - c_2^2)x'_B \equiv s^2 - s^1 \pmod q \tag{5}$$

From Eq. (4), we know that there is *at least* one $i \in \{1, 2\}$ such that $c_i^1 \not\equiv c_i^2 \pmod q$. Since $c_2^1 = c_2^2 = H(m\|L)^{x''_B} \bmod q$, which is the same for both simulations, it must be the case that $c_1^1 \not\equiv c_1^2 \pmod q$. Therefore, $\mathcal{S}$ can obtain x^* from Eq. (5).

Hence if the advantage of $\mathcal{F}$ in Game Unforgeability is ϵ, the probability that $\mathcal{S}$ solves a random DLOG problem instance is at least $(\epsilon/4)^2$ due to the forking lemma [14]. If each query takes at most time t_o to finish, the simulation time of the game is at most $2t + 2Qt_o + c$ where c denotes some constant time for system setup and key generation. □

Lemma 2 (Cheating Nominator). *For the NS scheme proposed above, given nominator A's private key, if a (t, ϵ, Q)-forger wins Game Unforgeability, there exists a $(t + Qt_o + c, \epsilon/Q)$-adversary which can solve the CDH (Computational Diffie-Hellman) problem in $\mathbb{Z}_q^*$, where t_o is the maximum time for simulating one oracle query and c is some constant.*

Proof. If a forger $\mathcal{F}$, after obtaining A's private key (x'_A, x''_A) via Corrupt, wins Game Unforgeability, we construct an algorithm $\mathcal{S}$ to solve a random CDH problem instance, that is, given $h, h^a, h^b \in \mathbb{Z}_q^*$, find $h^{ab} \in \mathbb{Z}_q^*$.

Game Simulation: At the beginning of the simulation of Game Unforgeability, $\mathcal{S}$ generates param according to SystemSetup, and sets nominee B's public key (y'_B, y''_B) to $(g^{x'_B} \bmod p, h^a \bmod q)$. For nominator A, the public/private key pair is generated using KeyGen accordingly. As in the proof of Lemma 1, a List is also maintained for random oracle H.

When $\mathcal{F}$ is invoked, 1^k, $pk_A = (y'_A, y''_A)$ and $pk_B = (y'_B, y''_B)$ are given to $\mathcal{F}$ and oracles are simulated as in the proof of Lemma 1. In particular, for a query to random oracle H, List is checked to see if the same query has made before, if so, the same answer will be returned; otherwise, $\mathcal{S}$ randomly picks $r \in_R \mathbb{Z}_{q-1}$ and sets the answer to $h^r \bmod q$. List is updated accordingly. As restricted, $\mathcal{F}$ cannot query Corrupt for B's private key.

Among the Q queries, $\mathcal{S}$ randomly chooses an H-query in the form $\tilde{m}\|pk_A\|pk_B$ and sets the answer to $h^b \bmod q$. In case that Confirmation/disavowal is queried on $(\tilde{m}, \sigma, pk_A, pk_B)$ for any nominative signature σ, the simulation aborts. The reason is that $\mathcal{S}$ does not know any of the discrete logarithms of y''_B and $H(\tilde{m}\|pk_A\|pk_B)$ which are a and b, respectively. In the following, we will see that at least $1/Q$ of the chance that this case will not happen.

Reduction: For one successful simulation, suppose the forgery of $\mathcal{F}$ is $\sigma^* = (s^*, c_1^*, c_2^*)$ on some message m^*, there must be an entry in List referring to $H(m^*\|y_A\|y_B) = h^{r^*} \bmod q$, by the assumption of random oracle model.

If $m^* = \tilde{m}$, we have $H(m^*\|pk_A\|pk_B) = h^b \bmod q$. Therefore, we have $c_2 = h^{ab} \bmod q$ which is the solution to the CDH problem instance. Since $\tilde{m}$ is randomly chosen, there is at least $1/Q$ chance that $m^* = \tilde{m}$. Therefore, $\mathcal{S}$ can obtain $h^{ab} \bmod q$ with probability at least ϵ/Q. Also note that $\mathcal{F}$ cannot query Confirmation/disavowal on $(m^*, \sigma, pk_A, pk_B)$ for any nominative signature σ. Hence the simulation will not have early abortion for the case that $m^* = \tilde{m}$.

If each query takes at most time t_o to simulate, the simulation time of the game is at most $t + Qt_o + c$ where c denotes some constant time for system setup and key generation. □

Theorem 1 (Unforgeability). *The NS scheme proposed above is unforgeable (Def. 1) if the DLOG problem in the subgroup of $\mathbb{Z}_p^*$ of order q generated by g and the CDH problem in $\mathbb{Z}_q^*$ are intractable.*

The theorem follows directly from Lemma 1 and 2.

Theorem 2 (Invisibility). *The NS scheme proposed above has the property of invisibility (Def. 2) under the Decisional Diffie-Hellman (DDH) assumption in the random oracle model. Precisely, if there exists a $(t, \frac{1}{2} + \epsilon, Q)$-distinguisher $\mathcal{D}$ in Game Invisibility, then there exists a $(t + Qt_o + c, \frac{1}{2} + \frac{\epsilon}{Q})$-distinguisher $\mathcal{D}^{DDH}$ which solves a DDH problem instance in $\mathbb{Z}_q^*$, where t_o is the maximum time for simulating one oracle query and c is some constant.*

Proof. If there exists a (t, ϵ, Q)-distinguisher $\mathcal{D}$ in Game Invisibility, we construct a DDH distinguisher $\mathcal{D}^{DDH}$ such that given $h, h^a, h^b, h^c \in \mathbb{Z}_q^*$, $\mathcal{D}^{DDH}$ is to decide if $c \equiv ab \pmod{q-1}$.

To simulate Game Invisibility, $\mathcal{D}^{DDH}$ follows the specification of the game and carries out a similar simulation to that in the proof of Lemma 2. In particular, $\mathcal{D}^{DDH}$ sets the second component of nominee B's public key y''_B to $h^a \bmod q$. As restricted, $\mathcal{D}$ cannot query Corrupt for B's private key. Another remark is that, before $\mathcal{D}$ outputs a challenge message m^*, $\mathcal{D}^{DDH}$ randomly chooses one of the H-queries that are in the form of $\tilde{m}\|pk_A\|pk_B$ and sets its answer to $h^b \bmod q$.

When $\mathcal{D}$ outputs a challenge message m^*, if $m^* \neq \tilde{m}$, $\mathcal{D}^{DDH}$ aborts and outputs a random bit b'; otherwise, $\mathcal{D}^{DDH}$ returns the challenge nominative signature $\sigma^* = (s^*, c_1^*, c_2^*)$ to $\mathcal{D}$. Since $\mathcal{D}^{DDH}$ knows x'_A, σ^* can be generated by $\mathcal{D}^{DDH}$ using SigGen, while setting c_2^* to $h^c \bmod q$. Note that $\mathcal{D}$ may have never queried H in the form of $\tilde{m}\|pk_A\|pk_B$ before outputting the challenge message m^*. In this case, $\mathcal{D}^{DDH}$ sets $H(m^*\|L)$ to $h^b \bmod q$ and sends to $\mathcal{D}$ the challenge nominative signature σ^* in the same way as the case when $m^* = \tilde{m}$. At the end of the game, $\mathcal{D}^{DDH}$ outputs whatever $\mathcal{D}$ outputs. Let $\mathbf{E}$ be the event that $\mathcal{D}^{DDH}$ does not abort when $\mathcal{D}$ outputs the challenge message m^*. Obviously, $\Pr[\mathbf{E}]$ is at least $1/Q$.

Note that $\sigma^* = (s^*, c_1^*, c_2^*)$ is always a valid ring signature on m^* with respect to $L = (pk_A, pk_B)$. As (h, h^a, h^b, h^c) is a DDH problem instance, half of the chance that $(h, y''_B, H(m^*\|L), c_2)$ is a DH-tuple and the other half that the instance is a non-DH-tuple. If (h, h^a, h^b, h^c) is a DH-tuple, so is $(h, y''_B, H(m^*\|L), c_2)$. Also note that $(m^*, pk_A, pk_B, role)$ cannot be queried to $\mathsf{SignTranscript}$, for any valid value of $role$, and $(m^*, \sigma, pk_A, pk_B)$ cannot be queried to $\mathsf{Confirmation/disoavowal}$ either, for any nominative signature σ. The simulation by $\mathcal{D}^{DDH}$ will not have early abortion.

Therefore, for event $\mathbf{E}$, if the probability that $\mathcal{D}$ wins $\mathsf{Game\ Invisibility}$ is $\frac{1}{2} + \epsilon$, $\mathcal{D}^{DDH}$ solves the DDH problem instance also with probability $\frac{1}{2} + \epsilon$. For event $\overline{\mathbf{E}}$, the probability that $\mathcal{D}^{DDH}$ solves the DDH problem instance is $\frac{1}{2}$ only. Therefore, the probability that $\mathcal{D}^{DDH}$ solves the DDH problem instance is equal to $\Pr[\mathbf{E}](\frac{1}{2} + \epsilon) + \Pr[\overline{\mathbf{E}}]\frac{1}{2}$. Since $\Pr[\mathbf{E}]$ is *at least* $1/Q$, the winning probability of $\mathcal{D}^{DDH}$ is at least $\frac{1}{2} + \frac{\epsilon}{Q}$. Similar to Lemma 2, the running time of $\mathcal{D}^{DDH}$ is at most $t + Qt_o + c$. □

Both confirmation and disavowal protocols in this scheme apply directly the technique due to Kurosawa and Heng [12]. Their technique has been proven to satisfy the requirement of non-transferability, which is defined in a similar way as the security against impersonation (Def. 2.3). In addition, by using the technique of Theorem 2, it can be shown that compromising the security against impersonation of this scheme reduces to compromising the underlying hard problem assumption of the confirmation/disavowal protocols of the undeniable signature scheme [2]. We skip the details but readers can readily derive the reduction from the proving technique of Theorem 2.

The scheme also satisfies the requirement that nominee cannot repudiate specified in Sec. 2.4. We refer readers to the remark at the end of Sec. 2.4 for the reason.

5 Conclusion

We proposed an efficient NS scheme. Unlike the only proven secure NS scheme available in the literature [13], which requires multi-round communications between the nominator and the nominee for signature generation, our construction requires only one-move communication. We also showed that the construction is proven secure under the strongest security model currently available and the reductionist proofs only relies on some standard number-theoretic assumptions. Our construction is also an interesting application of two-member ring signature. We believe that this new application of ring signature can inspire the construction of some other related cryptographic algorithms.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 415–432. Springer, Heidelberg (2002)
2. Chaum, D.: Zero-knowledge undeniable signatures. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 458–464. Springer, Heidelberg (1991)
3. Chaum, D.: Designated confirmer signatures. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 86–91. Springer, Heidelberg (1995)
4. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
5. Chaum, D., van Antwerpen, H.: Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (1992)
6. Chen, L., Kudla, C., Paterson, K.: Concurrent signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 287–305. Springer, Heidelberg (2004)
7. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proc. 22nd ACM Symp. on Theory of Computing, pp. 416–426 (May 1990)
8. Guo, L., Wang, G., Wong, D.: Further discussions on the security of a nominative signature scheme. Cryptology ePrint Archive, Report 2006/007 (2006)
9. Huang, Z., Wang, Y.: Convertible nominative signatures. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 348–357. Springer, Heidelberg (2004)
10. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
11. Kim, S.J., Park, S.J., Won, D.H.: Zero-knowledge nominative signatures. In: PragoCrypt 1996, International Conference on the Theory and Applications of Cryptology. pp. 380–392 (1996)
12. Kurosawa, K., Heng, S.: 3-move undeniable signature scheme. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 181–197. Springer, Heidelberg (2005)
13. Liu, D.Y.W., Wong, D.S., Huang, X., Wang, G., Huang, Q., Mu, Y., Susilo, W.: Nominative signature: Application, security model and construction. Cryptology ePrint Archive, Report 2007/069 (2007), http://eprint.iacr.org/2007/069
14. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
15. Rivest, R., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)

16. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, Cambridge (2005)
17. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Laih, C-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (2003)
18. Susilo, W., Mu, Y.: On the security of nominative signatures. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 329–335. Springer, Heidelberg (2005)

Anonymous Authentication Protocols with Credit-Based Chargeability and Fair Privacy for Mobile Communications[⋆,⋆⋆]

Chun-I Fan and Vincent Shi-Ming Huang

Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan
TEL.: +886-7-5252000 ext. 4346
FAX: +886-7-5254301
cifan@cse.nsysu.edu.tw

Abstract. Smart and tiny mobile phones are widely popularized and advanced mobile communication services are provided increasingly such that ubiquitous computing environments will come true soon. It is a pleasure for mobile users to work or get recreations in the mobile network environments. However, just as the cases in wireline networks, there are many security threats to mobile network systems and their impact on the security is more serious than that in wireline networks owing to the features of wireless transmissions and the ubiquity property in mobile network systems. The secret personal information, important data, or classified documents which mobile users carry may be stolen by malicious entities. In order to guarantee the quality of the advanced communication services, the security and privacy would be important issues when mobile users roam to the mobile networks. In this paper, an anonymous authentication scheme will be proposed to protect both the security of the mobile network system and the privacy of mobile users. Not only does the proposed scheme provide mutual authentication between each user and the system, but also the identity of each user can be kept secret against anyone else, including the system. Although the system anonymously authenticates the users, it can still make correct bills to charge these anonymous users. Finally, our protocols also achieve the goal of fair privacy which allows the judge to be able to revoke the anonymity and trace the illegal users when they misused the anonymity property such as they committed crimes.

Keywords: Mutual Authentication, Anonymity, Fair Privacy, Cryptography, Mobile Networks, Ubiquitous Computing.

[⋆] This research was partially supported by the National Science Council of the ROC (Taiwan) under grant NSC 95-2219-E-110-004.

[⋆⋆] This work was also supported in part by TWISC@NCKU under grant NSC 94-3114-P-006-001-Y.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 412–427, 2007.
© Springer-Verlag Berlin Heidelberg 2007

1 Introduction

Recently, mobile communication is becoming more and more popular such that many applications and services are provided in the mobile network environments. Moreover, some countries are planning to construct new wireless network architectures of 4G (4th Generation) mobile networks. There are also smart mobile equipments produced for people to enjoy the mobile services anywhere and anytime. It is obvious that mobile computing will penetrate the people's life in the near future. The convenient mobile network services and the powerful mobile equipments will make the people all around the world be willing to join the society of mobile communications.

Mobile users may process important documents or secret personal information in their mobile equipments when they roam around the mobile networks. They would worry about whether it is secure for them to carry their important data to the mobile networks. When mobile users exchange messages in the mobile networks, they will face lots of security threats. The eavesdroppers may try to obtain their communicating messages, their real identities, and even their locations where they are roaming around. The more information the eavesdroppers know, the less security and privacy the mobile users obtain. Sometimes the vicious insiders of the system operator would disclose the classified information of mobile users. Any system without maintaining user privacy will not be acceptable in the future.

There exist some weaknesses on user privacy in the existent 2G mobile network system. Each mobile user's alias, TMSI, can be linked to her/his real identity, IMSI, by attackers when the VLR requests her/him to retransmit her/his IMSI. The 2G mobile network also has no design for satisfying mutual authentication and protecting the users' privacy against the system operator. A mobile user may be cheated by some fake base stations in a mobile network system due to lack of mutual authentication. Although the 3G system has provided mutual authentication, the privacy or anonymity of mobile users has not been sufficiently considered yet.

Most of the proposed authentication schemes [5] [6] [7] [8] [9] [10] [12] which emphasize the privacy of mobile users usually assign an anonymous identity to each user. A mobile user will obtain an anonymous identity after she/he is authenticated by the system operator successfully, and she/he will take this valid alias to roam over the mobile networks. The eavesdroppers do not know the relation between her/his real identity and alias. To protect the user's privacy perfectly, we hope that anyone else, even the system operator, cannot derive such relations either. Owning to the unlinkability property, the technique of *blind signatures* [14] can help us with realizing complete anonymity for mobile users.

Another problem is that once a mobile user gets anonymity, how can the system operator charge her/him when she/he requests the mobile network services via an anonymous identity? Especially, if there is some mobile user who misused the anonymity property to commit crimes, how can the judge handle it? Almost all of the current solutions cannot cope with all of the above problems at the same time.

In our solution, every mobile user is anonymous from any other one's point of view when she/he is obtaining the mobile network services. Furthermore, the system operator can charge the mobile user according to the communication time the user consumed. Moreover, we also consider the issue of fair privacy. The privacy of the mobile users who misused the anonymity property can be revoked by the judge, and the police can trace the criminals who get anonymity. This is the property of *fair privacy*. We simultaneously realize the *anonymity*, *chargeability*, and *fair privacy* (revokeability and traceability) in our proposed authentication protocols for mobile communications.

2 Our Protocols

In order to simplify our protocols, we adopt the RSA cryptosystem [15] to realize them. However, our protocols can be implemented by any other suitable cryptosystem. First, we define and explain some notations as follows:

1. MS, H, V: These are three participants in our protocols. MS is a mobile user, H is the server of the home network, and V is the server of a visiting network.
2. ID_{MS}: The real identity of MS.
3. E_x: An encryption function where x can be a symmetric key or a public key.
4. $k_{ms\_h}, k_{ms\_v}, k_{v\_h}$: The shared session keys between MS and H, MS and V, and V and H, respectively.
5. pk_j, pk_v: The public keys of the judge and V, respectively.
6. F_1 and F_2: Two public one-way hash functions.
7. **A judge's device**: The judge issues a tamper-resistant device which contains {a random-number generator, a symmetric-key cryptosystem, a public-key cryptosystem, a public-private key pair of the judge, F_1, F_2}. This device will be integrated into the system of H. It is impossible to steal or modify any information embedded in the device. In our scheme, the judge is an off-line party, i.e., the judge does not need to keep connection with H in our protocols, but the judge's device does.
8. γ: It is a due date. As shown in Fig. 1, if a mobile user requests a ticket for communication in time slot P_i, H will assign her/him a due date γ where γ is the last day of next time slot P_{i+1}. H assigns the same γ to each mobile user who requests a ticket in the same time slot. The length of each time slot is equivalent.

Our scheme consists of four protocols which are described in Section 2.1, Section 2.2, Section 2.3, and Section 2.4, respectively. In our scheme, a mobile user requests an anonymous ticket by performing the protocol in Section 2.1. Then she/he can use the anonymous ticket for network services by executing the protocol in Section 2.2. After she/he performs the protocol in Section 2.2 for network services, H can charge her/him on the due date via the protocol in Section 2.3. Especially, if she/he does something illegal, the judge and the police can revoke her/his privacy or trace her/him through the protocol in Section 2.4.

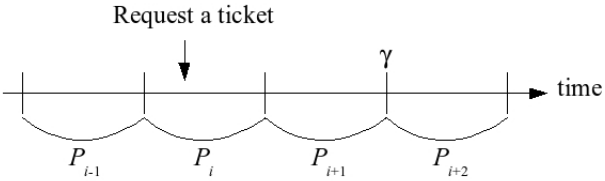


Fig. 1. The time slots

2.1 The Protocol for Requesting an Initial Anonymous Ticket

First, H chooses two distinct large primes p and q and computes $n = pq$. H also chooses its public key e and a secret key d such that $ed = 1 \bmod \phi(n)$ where $\phi(n) = (p-1)(q-1)$. Finally, H publishes $\{n, e\}$ and keeps $\{p, q, d\}$ secret. Besides, H also publishes all time slots P_is, $i \in \{1, 2, 3, ...\}$, i.e., all of the due dates γ are published.

In our scheme, the mobile user, MS, can request an anonymous ticket by running the protocol in this section after she/he performs any existing secure mutual authentication protocol with the system, V and H, where the secure mutual authentication protocol will establish the three session keys, k_{ms_h}, k_{ms_v}, and k_{v_h}, among MS, V, and H [2]. MS shares k_{ms_h} with H and k_{ms_v} with V, and V shares k_{v_h} with H. This protocol contains the following steps. The protocol is also shown in Fig. 2.

1. $MS \rightarrow H : \{E_{k_{ms\_h}}(M, F_1(M))\}$. First, MS randomly generates strings $\{m,\ k,\ v\}$ and an integer $r \in \mathbb{Z}_n^*$. Then MS computes $\alpha = r^e F_1^2(m) \bmod n$. Finally, MS forms $M = \{\alpha, E_{pk_j}(k, v, ID_{MS})\}$ and submits $E_{k_{ms\_h}}(M,$ $F_1(M))$ to H.
2. $H \rightarrow$ The judge's device : $\{\mu,\ \gamma,\ E_{pk_j}(k,\ v,\ ID_{MS})\}$. In this step, H knows that MS whose real identity is ID_{MS} wants to request a ticket. First, it sets $\mu = ID_{MS}$, and lets γ be the last day of next time slot. Then H inputs $\{\mu,$ $\gamma,\ E_{pk_j}(k, v, ID_{MS})\}$ into the judge's device. Finally, H records that MS has ever bought a ticket in the current time slot and she/he will have to return an unused ticket before the due date γ for billing.
3. The judge's device $\rightarrow H : \{\beta,\ E_k(\delta,\ b,\ v,\ \sigma),\ E_{pk_j}(r_z)\}$. First, the judge's device decrypts $E_{pk_j}(k,\ v,\ ID_{MS})$ and checks if $\mu = ID_{MS}$. If true, it randomly generates two strings (r_j, r_z) and an integer $b \in \mathbb{Z}_n^*$. Then it sets $w = 0$ and computes $\sigma = E_{pk_j}(w, r_j)$, $\delta = E_{pk_j}(ID_{MS}, F_1(r_z))$, and $\beta = b^e F_2(\delta, \sigma, \gamma) \bmod n$. Finally, it returns $\{\beta,\ E_k(\delta,\ b,\ v,\ \sigma),\ E_{pk_j}(r_z)\}$ to H.
4. $H \rightarrow MS : \{t, E_k(\delta, b, v, \sigma), \gamma\}$. First, H must record $(ID_{MS}, E_{pk_j}(r_z))$ into its database. Then H computes t=$(\alpha\beta)^d \bmod n$ and sends $\{t, E_k(\delta, b, v, \sigma), \gamma\}$ to MS.
5. **Unblinding:** After receiving $\{t,\ E_k(\delta,\ b,\ v,\ \sigma),\ \gamma\}$, MS checks if γ is the last day of next time slot. Then she/he decrypts $E_k(\delta,\ b,\ v,\ \sigma)$ via k and computes $s = (br)^{-1} t \bmod n$ after checking that v is the same as the one

which she/he sent to H previously. She/He obtains a ticket $(m, \delta, \sigma, \gamma, s)$ and can verify it by examining if the following formula is true.

$$F_1^2(m)F_2(\delta,\sigma,\gamma) \equiv s^e \pmod n \tag{1}$$

Finally, MS sets $i = 1$ and $(m_i, \delta_i, \sigma_i, \gamma, s_i) = (m, \delta, \sigma, \gamma, s)$ and then goes to the protocol of Section 2.2 when she/he decides to use the ticket.

2.2 The Protocol for Using an Anonymous Ticket at i-th Round Before the Due Date

This protocol makes it possible for the anonymous mobile user MS to perform mutual authentication with V and use her/his ticket for mobile network services. It contains the following steps and also be shown in Fig. 3.

1. $MS \rightarrow V : \{\theta, T, M\}$. First, MS sets $(m^*, \delta^*, \sigma^*, s^*) = (m_i, \delta_i, \sigma_i, s_i)$ and then prepares $T = \{F_1(m^*), \delta^*, \sigma^*, \gamma, s^*\}$ and randomly generates

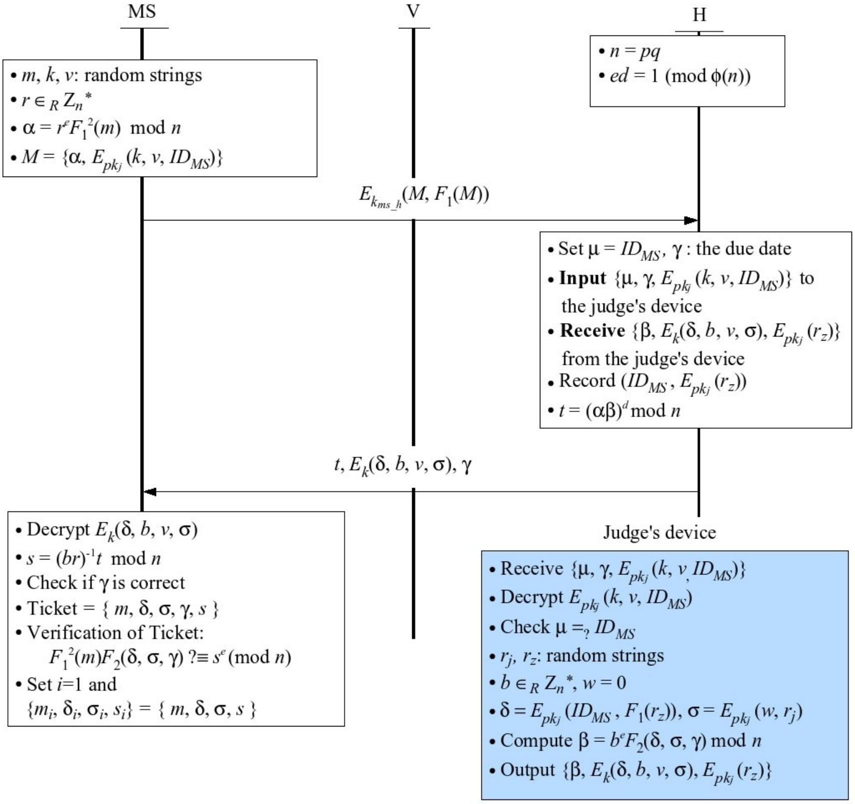

MS
V
H
• $n = pq$
• $ed = 1 \pmod{\phi(n)}$
• m, k, v: random strings
• $r \in_R Z_n^*$
• $\alpha = r^e F_1^2(m) \bmod n$
• $M = \{\alpha, E_{pk_j}(k, v, ID_{MS})\}$
$E_{k_{ms\_h}}(M, F_1(M))$
• Set $\mu = ID_{MS}$, γ : the due date
• **Input** $\{\mu, \gamma, E_{pk_j}(k, v, ID_{MS})\}$ to the judge's device
• **Receive** $\{\beta, E_k(\delta, b, v, \sigma), E_{pk_j}(r_z)\}$ from the judge's device
• Record $(ID_{MS}, E_{pk_j}(r_z))$
• $t = (\alpha\beta)^d \bmod n$
$t, E_k(\delta, b, v, \sigma), \gamma$
• Decrypt $E_k(\delta, b, v, \sigma)$
• $s = (br)^{-1}t \bmod n$
• Check if γ is correct
• Ticket = $\{ m, \delta, \sigma, \gamma, s \}$
• Verification of Ticket: $F_1^2(m)F_2(\delta, \sigma, \gamma) \mathrel{?}\equiv s^e \pmod n$
• Set i=1 and $\{m_i, \delta_i, \sigma_i, s_i\} = \{ m, \delta, \sigma, s \}$
Judge's device
• Receive $\{\mu, \gamma, E_{pk_j}(k, v, ID_{MS})\}$
• Decrypt $E_{pk_j}(k, v, ID_{MS})$
• Check $\mu =_? ID_{MS}$
• r_j, r_z: random strings
• $b \in_R Z_n^*, w = 0$
• $\delta = E_{pk_j}(ID_{MS}, F_1(r_z))$, $\sigma = E_{pk_j}(w, r_j)$
• Compute $\beta = b^e F_2(\delta, \sigma, \gamma) \bmod n$
• Output $\{\beta, E_k(\delta, b, v, \sigma), E_{pk_j}(r_z)\}$

Fig. 2. The Protocol for Requesting an Initial Anonymous Ticket

strings $\{m^{**}, k, v, r_1, r_2, r_3\}$ and an integer $r \in \mathbb{Z}_n^*$. Furthermore, MS computes $\alpha = r^e F_1^2(m^{**}) \bmod n$ and prepares $M = (\alpha, E_{pk_j}(k, v))$. Finally, MS computes $\theta = E_{pk_v}(r_1, r_2, r_3, F_1(M))$ and submits $\{\theta, T, M\}$ to V.

2. $V \rightarrow MS : \{r_1, r_4\}$. After receiving $\{\theta, T, M\}$, V verifies T by examining if
$$F_1(F_1(m^*))F_2(\delta^*, \sigma^*, \gamma) \equiv (s^*)^e \pmod{n} \tag{2}$$
and γ is not expired. If true, V decrypts θ and checks that the hashed value of M equals to $F_1(M)$. Then V generates a string r_4 randomly and sends $\{r_1, r_4\}$ to MS.
3. $V \rightarrow H : \{E_{k_{v\_h}}(T, F_1(T))\}$. After V sends $\{r_1, r_4\}$ to MS, it immediately submits $E_{k_{v\_h}}(T, F_1(T))$ to H in order to perform the double-usage checking on T. If T is doubly used, the connection will be terminated.
4. $MS \rightarrow V : \{\lambda\}$. After receiving $\{r_1, r_4\}$, MS checks if r_1 is the same as the one which was chosen by herself/himself. Then MS computes $\lambda = r_2 \oplus (m^*||r_4)$ and sends λ to V.
5. **Allowing Communication:** After receiving λ, V decrypts it by computing $(\lambda \oplus r_2)$ and parse the decryption result to get (m^*, r_4). Then, V verifies if r_4 equals to the one that was chosen by itself and checks whether the hashed value of m^* is the same as $F_1(m^*)$ in ticket T. If true, V ensures that MS is the real owner of T. Therefore, V allows MS to communicate. During the communication with V, MS can encrypt her/his messages via key r_3.
6. $V \rightarrow H : \{E_{k_{v\_h}}(m^*, M, w'), F_1(m^*, M, w')\}$. After MS terminated the communication, V computes the spent value w' of MS according to MS's communicating time or utilized services. Then V sends $E_{k_{v\_h}}(m^*, M, w')$ and $F_1(m^*, M, w')$ to H.
7. $H \rightarrow$ The judge's device: $\{\delta^*, \sigma^*, w', \gamma, E_{pk_j}(k, v)\}$. Once H receives the spent value w' of MS, H stores $\{T, m^*, w'\}$ into its database. Then H extracts δ^*, σ^*, and γ from T and extracts $E_{pk_j}(k, v)$ from M. Finally, H inputs $\{\delta^*, \sigma^*, w', \gamma, E_{pk_j}(k, v)\}$ into the judge's device.
8. The judge's device $\rightarrow H : \{\beta, E_k(\delta^{**}, b, v, \sigma^{**})\}$. First, the judge's device decrypts $E_{pk_j}(k, v)$ to get (k, v) and parses the result of decrypting δ^* as (ID_{MS}, z'). It also decrypts σ^* to get the first element, named w^*, in the decrypting result. Furthermore, it randomly selects a string r_j and an integer $b \in \mathbb{Z}_n^*$. Then it prepares $\delta^{**} = E_{pk_j}(ID_{MS}, F_2(z'))$ and $\sigma^{**} = E_{pk_j}((w^* + w'), r_j)$. Finally, it computes $\beta = b^e F_2(\delta^{**}, \sigma^{**}, \gamma) \bmod n$ and outputs $\{\beta, E_k(\delta^{**}, b, v, \sigma^{**})\}$ to H.
9. $H \rightarrow MS : \{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$. H computes $t = (\alpha\beta)^d \bmod n$ and returns $\{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$ to MS.
10. **Unblinding:** After receiving $\{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$, MS decrypts the ciphertext $E_k(\delta^{**}, b, v, \sigma^{**})$ and checks v, and then MS computes $s^{**}{=}(br)^{-1}t \bmod n$. Finally, MS obtains a new ticket $(m^{**}, \delta^{**}, \sigma^{**}, \gamma, s^{**})$ which can be verified by checking whether $F_1^2(m^{**})F_2(\delta^{**}, \sigma^{**}, \gamma) \equiv (s^{**})^e \pmod{n}$ is true or not. If true, MS sets $i = i{+}1$ and $(m_i, \delta_i, \sigma_i, \gamma, s_i) = (m^{**}, \delta^{**}, \sigma^{**}, \gamma, s^{**})$, which is an unused ticket of the user. Thus, she/he can use the new unused ticket for the next round of communication before the due date, γ.

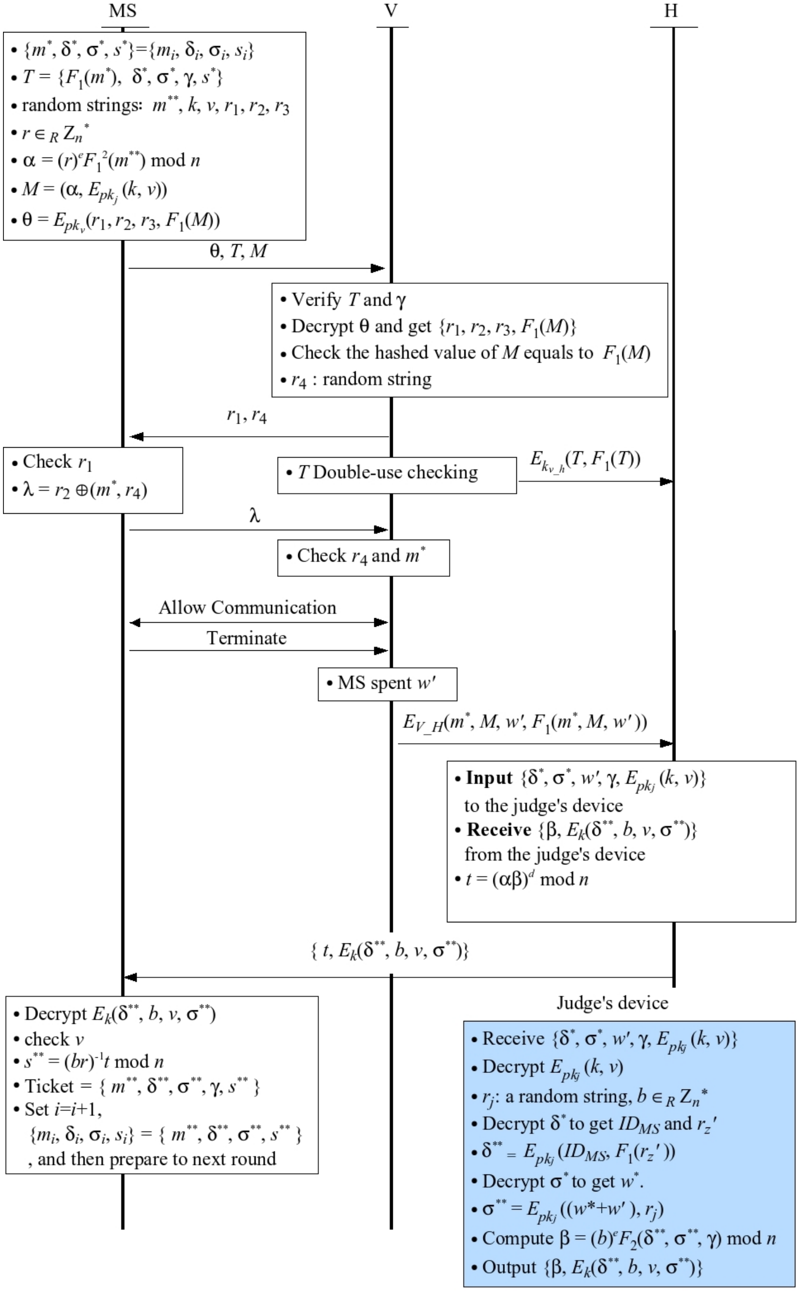

MS
V
H
• $\{m^*, \delta^*, \sigma^*, s^*\}=\{m_i, \delta_i, \sigma_i, s_i\}$
• $T = \{F_1(m^*), \delta^*, \sigma^*, \gamma, s^*\}$
• random strings: $m^{**}, k, v, r_1, r_2, r_3$
• $r \in_R Z_n^*$
• $\alpha = (r)^e F_1^2(m^{**}) \bmod n$
• $M = (\alpha, E_{pk_j}(k, v))$
• $\theta = E_{pk_v}(r_1, r_2, r_3, F_1(M))$
θ, T, M
• Verify T and γ
• Decrypt θ and get $\{r_1, r_2, r_3, F_1(M)\}$
• Check the hashed value of M equals to $F_1(M)$
• r_4 : random string
r_1, r_4
• Check r_1
• $\lambda = r_2 \oplus (m^*, r_4)$
• T Double-use checking
$E_{k_{v\_h}}(T, F_1(T))$
λ
• Check r_4 and m^*
Allow Communication
Terminate
• MS spent w'
$E_{V\_H}(m^*, M, w', F_1(m^*, M, w'))$
• **Input** $\{\delta^*, \sigma^*, w', \gamma, E_{pkj}(k, v)\}$ to the judge's device
• **Receive** $\{\beta, E_k(\delta^{**}, b, v, \sigma^{**})\}$ from the judge's device
• $t = (\alpha\beta)^d \bmod n$
$\{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$
• Decrypt $E_k(\delta^{**}, b, v, \sigma^{**})$
• check v
• $s^{**} = (br)^{-1} t \bmod n$
• Ticket = $\{m^{**}, \delta^{**}, \sigma^{**}, \gamma, s^{**}\}$
• Set i=i+1,
$\{m_i, \delta_i, \sigma_i, s_i\} = \{m^{**}, \delta^{**}, \sigma^{**}, s^{**}\}$
, and then prepare to next round
Judge's device
• Receive $\{\delta^*, \sigma^*, w', \gamma, E_{pkj}(k, v)\}$
• Decrypt $E_{pkj}(k, v)$
• r_j: a random string, $b \in_R Z_n^*$
• Decrypt δ^* to get ID_{MS} and r_z'
• $\delta^{**} = E_{pkj}(ID_{MS}, F_1(r_z'))$
• Decrypt σ^* to get w^*.
• $\sigma^{**} = E_{pkj}((w^*+w'), r_j)$
• Compute $\beta = (b)^e F_2(\delta^{**}, \sigma^{**}, \gamma) \bmod n$
• Output $\{\beta, E_k(\delta^{**}, b, v, \sigma^{**})\}$

Fig. 3. The Protocol for Using an Anonymous Ticket at i-th Round Before the Due Date

2.3 The Protocol for Charging Mobile Users

For each mobile user, say MS, the system operator, H, audits her/his bill through the following steps on the due date, γ:

1. MS returns her/his real identity and unused ticket, $(m^*, \delta^*, \sigma^*, \gamma, s^*)$, to H on the due date.
2. H checks that the ticket does not exist in its database, and then extracts σ^* from the ticket. H sends σ^* to the judge's device.
3. The judge's device decrypts σ^* and returns the first element in the decryption result i.e., the spent value, to H.
4. H adds the spent value to the bill of MS and deletes the record which indicates that MS has ever bought a ticket.
5. Send the bill to MS.

Besides, if the mobile user wants to request a ticket after the due date, γ, she/he must perform the protocol of Section 2.1 again.

Our scheme adopts credit-based charging[1], just as the practical situation in the real world. It is different from the others which provided approaches of debit-based charging[2][5][12]. What are the differences between charging mobile users in advance and charging them later? The followings are the reasons why we design our scheme to charge mobile users later.

1. **Adaptability.** In current GSM services, almost all of the systems adopt credit-based ways to charge users.
2. **Reducing the relations between any two rounds of communication with one token only.** There are two possible ways to charge a mobile user in advance, which are described as follows:
   (a) The mobile user purchases a set of payment tokens from the system previously where each of the tokens is with a unit value. In each round of communication, the mobile user sends a proper number of tokens to the system for payment. In this case, it is difficult for the system to derive the relation between any two rounds of communication since each token is independent. However, this will consume much storage and space for recording and transmitting these tokens.
   (b) The mobile user purchases only one payment token from the system previously where the token is with a specific value w. In the following round of communication, the mobile user sends the token to the system for payment and then the system returns a token with value $(w - w_1)$ if the user consumes w_1 value of that token. The mobile user just needs to store one token. However, this will cause defective privacy. When the system returns one token with value $(w - w_1)$ to the user, the system knows that the user will use the token with value $(w - w_1)$ in the next round. There exists a relation between these two rounds.

[1] The system charges each mobile user after she/he used the mobile network services.

[2] The system charges each mobile user before she/he starts to use the mobile network services.

In our scheme, we greatly reduce the relations between any two rounds of communication from the system's point of view with one-token storage only.

3. **Free from the problem of overspending.** In debit-based charging methods, both of the above two ways (a) and (b), when a mobile user shows her/his token(s) to the system for communicating, her/his communication will be terminated if the tokens or the token's value are used up. It will cause inconvenience for the mobile user. If the system does not terminate the communication, the mobile user will overspend the token(s) and the system must perform extra procedures to deal with the situation. In our scheme, based on a credit-based method, the above problem can be avoided.

2.4 The Protocol for Privacy Revoking

In some special situations, H or the judge needs to disclose the identity of an anonymous mobile user. For example, an anonymous mobile user commits a crime; the police wants to trace some criminals; or some mobile users who do something harmful for H. Our scheme supports two ways to trace illegal anonymous mobile users.

1. **Tracing the mobile user by a designated ticket:** Once a mobile user impose on anonymity to commit a crime, her/his ticket will be reported to the judge. Assume that the ticket is $(m', \delta', \sigma', \gamma', s')$. The judge will extract δ' from the ticket and parse the decryption result of δ' to get ID_{MS}.
2. **Tracing the tickets by a designated mobile user:** When the police wants to trace a criminal whose real identity is ID_{MS}, the police will ask H and the judge to disclose the privacy of the criminal. In this case, H will send the recorded pair $(ID_{MS}, E_{pk_j}(r_z))$ to the judge. After decrypting $E_{pk_j}(r_z)$, the judge will compute

$$\begin{cases} \delta_1 = E_{pk_j}(ID_{MS}, F_1^1(r_z)) \\ \delta_2 = E_{pk_j}(ID_{MS}, F_1^2(r_z)) \\ \delta_3 = E_{pk_j}(ID_{MS}, F_1^3(r_z)) \\ \vdots \\ \delta_i = E_{pk_j}(ID_{MS}, F_1^i(r_z)). \end{cases} \tag{3}$$

Then, it sends $\{\delta_1, \delta_2, \delta_3, \dots, \delta_i\}$ to H, and H will help the police to trace the mobile user via the set. In our scheme, the mobile user uses T_1 with δ_1 for her/his first round of communication, T_2 with δ_2 for her/his second round, and so forth. According to this order, H can trace the communication activities of the criminal from the first round to the i-th round via $\{\delta_1, \delta_2, \delta_3, \dots, \delta_i\}$.

2.5 Exceptions

In addition to the above issues, there are still two exceptions that may happen in our scheme. One is that the mobile user denies returning her/his ticket for billing before the due date, and the other one is that the mobile user lost her/his ticket (or lost her/his mobile equipment).

1. **The mobile user denies returning her/his ticket for billing before the due date:** After the due date γ, if there is any mobile user who has not returned her/his unused ticket yet, H will send a list $\mathscr{L}$ to the judge where $\mathscr{L}$ contains the identities of the mobile users who did not return their unused tickets. According to $\mathscr{L}$, the judge sends payment notices to all of the mobile users on $\mathscr{L}$ and announces another due date $\gamma^{'}$. If there is a mobile user, ID_{MS}, who does not return her/his unused ticket before the new due date $\gamma^{'}$, the judge will compute the set $\{\delta_1, \delta_2, \delta_3, \ldots\}$ according to ID_{MS} by (3) and then sends it to H. Assume that the mobile user denied returning T_{i+1}. H can find $\{T_i, w'_i\}$ from its database via δ_i. When H finds $\{T_i, w'_i\}$, the judge can help H with extracting the spent value w^* from T_i, and then H computes $w'' = w^* + w'_i$ and adds w'' to the bill of ID_{MS}.
2. **The mobile user lost her/his ticket:** When a mobile user lost her/his unused ticket T_i, she/he must ask H to freeze her/his unused ticket or it may be used by an attacker. After an authorization process[3] where the mobile user authorizes H to reveal her/his privacy, H can send $(ID_{MS}, E_{pk_j}(r_z))$ to the judge to compute $\{\delta_1, \delta_2, \ldots\}$ by (3). Assume that the mobile user lost T_i, H must deny the services for $T_i, T_{i+1}, T_{i+2}, \ldots$ by $\delta_i, \delta_{i+1}, \delta_{i+2}, \ldots$, respectively, where $i \in \mathbb{N}$. Besides, H finds $\{T_{i-1}, w'_{i-1}\}$ from its database via δ_{i-1} and sends T_{i-1} to the judge to extract the spent value w^* from T_{i-1}. After the judge returns w^* to H, H adds $(w'_{i-1} + w^*)$ to the bill of the mobile user.

   In order to handle this exception, the privacy of $T_1, T_2, T_3, \ldots, T_{i-1}$ of the mobile user will be revealed. However, if the mobile user remembered how many tickets she/he has used, she/he can still preserve her/his privacy. For example, a mobile user lost her/his unused ticket, and she/he remembers that she/he has consumed 4 tickets. Then the judge just needs to compute $\{\delta_5, \delta_6, \ldots\}$ for H, and $\{\delta_1, \delta_2, \delta_3, \delta_4\}$ are still kept secret for the mobile user.

   After the mobile user freezes her/his lost ticket, she/he can perform the protocol in Section 2.1 again to request a new ticket.
3. **The communications are terminated abnormally:** Consider the case that the communication of Step 9 in Section 2.2 are abnormally terminated, i.e., the mobile user does not receive a renewed ticket after communicating. We assume that each time when the mobile user receives $\{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$ successfully, she/he will return an ACK to H. Once H does not receive ACK, it will store $\{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$ and $\{m^*, \delta^*, \sigma^*, \gamma, s^*\}$ into an unsuccessful communication record. Thus, the user can retransmit $\{m^*, \delta^*, \sigma^*, \gamma, s^*\}$ to H via V, and H can retrieve $\{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$ from the unsuccessful communication record and send it to the mobile user.

[3] For example, the mobile user must sign a document to show that she/he agrees H to ask the judge to compute $\{\delta_1, \delta_2, \ldots\}$.

3 Security Analysis

In the protocols of Section 2.1 and Section 2.2, we adopt the technique of blind signatures to design the mechanisms for ticket requesting and ticket renewal, respectively. The protocol of Section 2.2 shows that an anonymous mobile user has to mutually authenticate with the system before accessing network services. In this section, we analyze several attacks on the above protocols.

3.1 The Replay Attack

If the system and a mobile user cannot detect replayed messages between their transmissions, the replay attack will be successful. In this subsection, we analyze the replay attack on the anonymous authentication steps of the protocol in Section 2.2. We will demonstrate that it will be detected by V or MS if any of their transmissions is replayed.

<**1**> $MS \rightarrow V : \{\theta, T, M\}$. If the message is replayed, the attacker cannot pass the authentication successfully because that the attacker cannot show m^* at the following transmission. Only the real owner of T knows the corresponding m^*.

<**2**> $V \rightarrow MS : \{r_1, r_4\}$. If there is a fake V which wants to cheat m^* out of MS, it will be detected by MS because of the random string r_1. The fake V cannot answer correct r_1 to MS since it cannot decrypt θ to get r_1 which was randomly chosen by MS.

<**3**> $MS \rightarrow V : \{\lambda\}$. If an attacker replays λ directly, it will be detected by V according to the random strings r_4 and r_2. The strings r_4 and r_2 are chosen randomly in each round. Even if the attacker replays $\{\theta, T, M\}$ and $\{\lambda\}$ simultaneously, it will be detected by V because of the random string r_4.

3.2 The Impersonation Attack

It is easy to duplicate a T of MS from the transmissions. However, it is computationally infeasible to use the duplicated T to impersonate MS in our protocol.

<**1**> **Impersonating** MS: V authenticates MS by verifying her/his ticket T_i and checking the secret m_i of T_i. Only the real owner has the corresponding m_i. If the hash function F_1 is secure enough and MS keeps its m_i secret, anyone else cannot impersonate MS.

<**2**> **Impersonating** V: Similarly, the attacker cannot impersonate V. A fake V will be detected by MS since it cannot send the correct r_1 to MS. As long as the public-key cryptosystem is secure and V keeps the secret key safely, the attacker cannot decrypt θ and then return correct r_1 to MS.

<**3**> **Impersonating** H: In the protocol of Section 2.1 or Section 2.2, if an attacker wants to impersonate H, it must return correct $\{t, E_k(\delta, b, v, \sigma)\}$ or $\{t, E_k(\delta^{**}, b, v, \sigma^{**})\}$ to MS. However, in our protocols when requesting or using a ticket, MS always sends H a challenge string v which is encrypted by E_{pk_j}. None can return correct v because that only the legal H has the judge's device which can help H to decrypt $E_{pk_j}(k, v, ID_{MS})$ or $E_{pk_j}(k, v)$.

3.3 Message Replacement

If an attacker replaces a message which is prepared for the system to sign and the replacement causes that the legal user cannot get a valid ticket or the legal user cannot check it out, then the attack will work. We examine the following transmissions to check whether the replacement attack can work or not.

<**1**> $MS \rightarrow H : \{E_{k_{ms\_h}}(M, F_1(M))\}$. In the protocol of Section 2.1, if the transmission is replaced, it will be detected by H after H decrypts the message and checks the hashed value of M. Even if the attacker replaces this transmission by another valid $E_{k_{ms\_h}}(M', F_1(M'))$, the mobile user will detect the replacement when she/he is verifying the received signature.

<**2**> $H \rightarrow MS : \{t, E_k(\delta, b, v, \sigma), \gamma\}$. In the protocol of Section 2.1, if any parameter of this transmission is replaced, it will be detected by MS via the verification formula (1). Even if t and $E_k(\delta, b, v, \sigma)$ are replaced simultaneously by another valid t', δ' and σ', MS can still detect it via v which is chosen by herself/himself.

<**3**> $MS \rightarrow V : \{\theta, T, M\}$. In the protocol of Section 2.2, θ contains $F_1(M)$. Hence, attackers must replace θ and M at the same time or H will detect the replacement. However, once an attacker replaces θ and M, it will be detected by MS because of the incorrect r_1.

<**4**> $V \rightarrow H : \{E_{v\_h}(m^*,\ M,\ w'),\ F_1(m^*, M, w')\}$. In the protocol of Section 2.2, if the transmission is replaced by another $\{E_{v\_h}(m^{*'},\ M',\ w''),$ $F_1(m^{*'}, M', w'')\}$, H will detect it through the value of $F_1(m^*)$.

<**5**> $H \rightarrow MS : \{t,\ E_k(\delta^{**},\ b,\ v,\ \sigma^{**})\}$. In the protocol of Section 2.2, if any parameter of this transmission is replaced, it will be detected by MS via an incorrect v.

3.4 The Security Requirements for Each Entity

In this section, we discuss the security requirements for mobile users, the judge, and the system, respectively. We show that why our protocol satisfies these requirements.

The Viewpoints of Mobile Users: From the viewpoints of a mobile user, what she/he would like to ensure are:

1. **Mutual authentication**: MS and V can be mutually authenticated by each other via Step 1, Step 2, and Step 4 of the protocol in Section 2.2. MS authenticates V by verifying if it sends a correct string r_1 where V must decrypt $E_{pk_v}(r_1)$ to obtain r_1. And MS shows the secret m^* after she/he ensures that the V is legal. Once MS shows her/his m^* to V, MS convince V that she/he is a legal user.
2. **None can impersonate** MS: According to <1> and <3> in Section 3.1 and <1> in Section 3.2, none can impersonate MS via the replay or impersonation attacks.

3. **None can impersonate the system**: According to <2> in Section 3.1 and <2> in Section 3.2, none can impersonate V.
4. **Ticket correctness**: When MS received a ticket in Step 5 of Section 2.1, she/he can verify whether the received ticket is correct or not by formula (1). Besides, <1>, <2>, <3>, <4>, and <5> in Section 3.3 show that the destruction of the ticket can be detected.
5. **Tickets would not be stolen when being transmitted**: In our scheme, MS does not have to worry that her/his T will be stolen as long as she/he keeps m^* secretly since <1> in Section 3.2 shows that none can impersonate MS by duplicating her/his T, and <2> in Section 3.2 demonstrates that none can impersonate V to cheat m^* out of MS.
6. **Obtaining privacy**: In Step 1, Step 2, and Step 4 of Section 2.2, MS does not need to reveal her/his identity to V for authenticating. Furthermore, each T of MS is independent, so that the system operator cannot trace MS via the information of T in each round. Furthermore, it is difficult for the system to link any two rounds of communication of the same user. Since each ticket is independent, just as the case of (a) in Section 2.3
7. **Secure communication**: In Step 5 of Section 2.2, MS can communicate with V via the session key r_3.

The Viewpoints of the System Operator: From the viewpoints of the system, including H and V, what it would like to ensure are:

1. **Malicious mobile users cannot pass the authentication:** In Section 2.2, V verifies T of MS in Step 1 and checks the corresponding m^* of T in Step 5. Besides, <1> and <3> of Section 3.1 also show that the system operator can also withstand the replay attack.
2. **None can impersonate the system operator:** According to <2> in Section 3.1, and <2> and <3> in Section 3.2, none can impersonate the system operator to cheat a mobile user. And by <2> and <5> in Section 3.3, none can impersonate the system to issue tickets.
3. **Charge anonymous mobile users correctly:** From Section 2.3, the system operator can charge mobile users correctly in normal cases. Even in the exception cases described in Section 2.5, the system operator can still charge the mobile users correctly.

The Viewpoints of the Judge: In the mobile network environments, the judge needs a mechanism to revoke the privacy of the mobile users who misused the anonymity property. It also needs the capability of tracing a criminal who roams over the mobile networks. In our scheme, the judge can supervise the illegal mobile users and criminals by the two ways shown in Section 2.4.

4 Comparisons

First, we describe some features as follows where these features are required for mobile users when they roam around the mobile networks.

1. **Hiding identity:** Mobile users hide their real identities from the system operator, H and V, and the eavesdroppers, E.
2. **No relation:** It is difficult for the system to derive the relation between any two rounds of the communication of the same mobile user.
3. **Secure channels:** After performing mutual authentication between an anonymous mobile user and the system operator, they must establish a shared session key for the following communication activities.
4. **Fair privacy:** Fair privacy contains **traceability** and **revokability**. If a crime happens, the police can trace the identities of related anonymous mobile users or the judge can revoke the privacy of a criminal mobile user.
5. **Chargeability:** Chargeability contains two methods, **debit-based** and **credit-based** methods. By the debit-based method, the system operator charges mobile users before they access mobile network services. By the credit-based method, the system operator charges mobile users later. Both of the two methods must be able to charge anonymous mobile users according to the time of communication or the types of services they utilized.

The comparisons of our proposed scheme and the others are summarized in Table 1. In Table 1, the authors of [5] also mentioned untraceability and revokability, but they did not realize it in their scheme. We believe that realizing untraceability and revokability is not trivial. Finally, as the mentioned in Section 2.3, the credit-based charging method is better than the debit-based one since

Table 1. Comparisons

	Privacy					**Property**		
	Hide Identity			NoR	S	Fair		Chargeability
	H	V	E			T	R	
ours	○	○	○	○	○	○	○	Credit-based
[5]	○	○	○	×	○	△	△	Debit-based
[6]	○	○	○	×	×	×	×	–
[7]	×	○	○	○	○	×	×	–
[8]	×	×	○	○	○	×	×	–
[9]	×	○	○	×	○	×	×	Credit-based
[10]	○	○	○	○	×	×	×	–
[12]	○	○	○	×	×	×	×	Debit-based
[16]	×	○	○	×	–	×	×	Debit-based
[17]	×	○	○	×	○	×	×	Debit-based

H: Home domain. **V**: Visiting domain. **E**: Eavesdroppers.
NoR: Hard to derive relation between any two rounds.
S: Secure Channel.
T: Traceability. Trace the criminal user.
R: Revokability. Revoke the user's privacy.
○: Achieving the feature.
×: Not achieving the feature.
△: No realization on the feature.
–: No consideration

the former **1) is the same as the practical situation in current GSM services; 2) can greatly reduce the relations between any two rounds of communication** and **3) is free from the problem of overspending**.

5 Conclusions

We have proposed a mobile authentication scheme which can authenticate mobile users anonymously. When a mobile user enters the anonymity mode, she/he can perform a mutual authentication process with the system operator. The system operator can charge the anonymous user correctly according to the time she/he consumed by a credit-based method. Furthermore, if some mobile user misuses the anonymity property, the judge can revoke her/his privacy and trace her/him.

In this paper, the privacy of an honest mobile user might be broken by the system operator if the mobile user lost her/his ticket since the system operator needs to trace her/his used tickets in order to find the spending value of her/him. It is an interesting research topic to find an efficient solution to cope with the problem.

Acknowledgments

We would like to thank the anonymous reviewers of this paper for their valuable comments.

References

1. Samfat, D., Molva, R., Asokan, N.: Untraceability in mobile networks. In: International Conference on Mobile Computing and Networking, pp. 26–36 (1995)
2. Hwang, K.F., Chang, C.C.: A self-encryption mechanism for authentication of roaming and teleconference services. IEEE Transactions on Wireless Communications 2(2), 400–407 (2003)
3. Asokan, N.: Anonymity in a mobile computing environment. Mobile Computing System and Applications, 200–204 (December 1994)
4. Ozturk, C., Zhang, Y., Trappe, W., Ott, M.: Source-location privacy for networks of energy-constrained sensors. In: Proceedings of Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, pp. 68–72 (May 2004)
5. Karygiannis, A., Kiayias, A., Tsiounis, Y.: A solution for wireless privacy and payments based on e-cash. Security and Privacy for Emerging Areas in Communications Networks, 206–218 (September 2005)
6. He, Q., Wu, D., Khosla, P.: The quest for personal control over mobile location privacy. Communications Magazine, IEEE, 130–136 (May 2004)
7. Park, S.Y., Han, M.S., Eom, Y.I.: An efficient authentication protocol supporting privacy in mobile computing environments. In: 5th IEEE International Conference on High Speed Networks and Multimedia Communications, pp. 332–334 (July 2002)
8. Zhu, J., Ma, J.: A new authentication scheme with anonymity for wireless environments. IEEE Transactions on Consumer Electronics 50(1), 231–235 (2004)

9. Kesdogan, D., Fouletier, X.: Secure location information management in cellular radio systems. Wireless Communication System Symposium, IEEE, 35–40 (November 1995)
10. Lin, W.D., Jan, J.K.: A wireless-based authentication and anonymous channels for large scale area. In: Proceedings of Sixth IEEE Symposium on Computers and Communictions, pp. 36–41 (July 2001)
11. Schneier, B.: Applied cryptography second edition: protocols, algorithms, and source code in C. Professional, Reference and Trade Group (1996)
12. Tracz, R., Wrona, K.: Fair electronic cash withdrawal and change return for wireless networks. In: International Workshop on Mobile Commerce, pp. 14–19 (2001)
13. Jakobsson, M., Yung, M.: Revokable and versatile electronic money. In: Conference on Computer and Communications Security, pp. 76–87 (1996)
14. Chaum, D.: Blind Signature Systems. In: Advances in Cryptology – CRYPTO 1983, p. 153. Plenum
15. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of ACM 21(2), 120–126 (1978)
16. Mu, Y., Varadharajan, V.: A new scheme of credit based payment for electronic commerce. In: 23rd Annual IEEE International Conference on Local Computer Networks, pp. 278–284 (October 1998)
17. Yang, C.-C., Tang, Y.-L., Wang, R.-C., Yang, H.-W.: A secure and efficient authentication protocol for anonymous channel in wireless communications. Applied Mathematics and Computation 169, 1431–1439 (2005)
18. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

How to Find Many Collisions of 3-Pass HAVAL

Kazuhiro Suzuki[1] and Kaoru Kurosawa[2]

[1] Venture Business Laboratory, Ibaraki University, Hitachi, Ibaraki 316-8511, Japan
[2] Department of Computer and Information Sciences, Ibaraki University, Hitachi, Ibaraki 316-8511, Japan

Abstract. The hash function HAVAL is a well known Merkle-Damgård hash function such as MD4 and MD5. It has three variants, 3-, 4- and 5-pass HAVAL. On 3-pass HAVAL, the best known attack finds a collision pair with 2^7 computations of the compression function. To find k collision pairs, it requires 2^7k computations. In this paper, we present a better collision attack on 3-pass HAVAL, which can find k collision pairs with only $2k + 33$ computations. Further, our message differential is different from the previous ones. It is important to find collisions for different message differentials.

Keywords: hash function, HAVAL, collision, differential attack.

1 Introduction

The hash function HAVAL was proposed by Zheng, Pieprzyk, and Seberry at Auscrypt '92 [8]. It is a well known Merkle-Damgård hash function such as MD4 and MD5.[1] HAVAL has three variants, 3-, 4- and 5-pass HAVAL, which means that the compression function has 96, 128, and 160 rounds, respectively. The compression function H of HAVAL takes a 256-bit initial value and a 1024-bit message $M = (m_0, \ldots, m_{31})$ as input, and produces 256-bit hash value as output, where each m_i is a 32-bit word.

On 3-pass HAVAL, Rompay, et al. [2] presented a collision attack that requires 2^{29} computations of the compression function. Their attack can find a one-block (1024-bit) collision pair $M = (m_0, \ldots, m_{31})$ and $M' = (m'_0, \ldots, m'_{31})$ with the differential

$$\Delta m_{28} = 2^0 = 1 \text{ and } \Delta m_i = 0 \text{ for the other } i.$$

X.Y.Wang et al. [4] showed a much better collision attack with 2^7 computations of the compression function. Their attack can find a one-block collision pair $M = (m_0, \ldots, m_{31})$ and $M' = (m'_0, \ldots, m'_{31})$ with the differential

$$\Delta m_0 = 2^{10},\ \Delta m_{11} = 2^{31},\ \Delta m_{18} = 2^3, \text{ and } \Delta m_i = 0 \text{ for the other } i.$$

[1] The newest version is HAVAL 1.1. One can download the program source code at the website [1]. The difference between the first version and HAVAL 1.1 is only the order of initial values and the other constant values.

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 428–443, 2007.

© Springer-Verlag Berlin Heidelberg 2007

To find k collision pairs of 3-pass HAVAL, the best known attack [4] requires 2^7k computations.

In this paper, we present a better collision attack on 3-pass HAVAL which can find k collision pairs with only $2k + 33$ computations. In other words, we are interested in finding many collision pairs while the previous works focused on finding a single collision pair. This is important because if we have a large set of collision pairs, it will be easier to find a meaningful message.

Further, our message differential is different from the previous ones.[2] It is important to find collisions for different message differentials.

The previous attacks [2,4] are one-block collision attacks (i.e. a collision pair is a pair of 1024-bit message block). On the other hand, our attack is a two-block collision attack which can find a two-block (2048-bit) collision pair $M_0||M_1 = (m_{0,0}, \ldots, m_{0,31}, m_{1,0}, \ldots, m_{1,31})$ and $M'_0||M'_1 = (m'_{0,0}, \ldots, m'_{0,31}, m'_{1,0}, \ldots, m'_{1,31})$ with the differential

$$\Delta m_{j,i} = m'_{j,i} - m_{j,i} \bmod 2^{32} = \begin{cases} 2^{31} & \text{if } i = 5, \\ 0 & \text{otherwise.} \end{cases}$$

In our attack, we first find a near-collision pair (M_0, M'_0) such that $H(M_0)$ and $H(M'_0)$ are almost the same. We then find many full collision pairs $(M_0||M_1, M'_0||M'_1)$ by using the freedom of (M_1, M'_1). Theoretically, our near-collision pair can be found by about 33 computations of the compression function. Once a near collision pair is found, a full collision pair can be found with probability 1/2. Hence we can find k collision pairs with $2k + 33$ computations. (See Table 1.)

Table 1. Collision attacks on 3-pass HAVAL

	Rompay, et al. [2]	X.Y.Wang et al. [7]	Proposed
Δm_i	$\Delta m_{28} = 2^0 = 1$	$\Delta m_0 = 2^{10}$ $\Delta m_{11} = 2^{31}$ $\Delta m_{18} = 2^3$	$\Delta m_{0,5} = 2^{31}$ $\Delta m_{1,5} = 2^{31}$
complexity for first collision	2^{29}	2^7	$2 + 33$
complexity for k collision pairs	$2^{29}k$	2^7k	$2k + 33$
message length	1024 bits	1024 bits	2048 bits

In our personal computer simulation (CPU:AthlonXP 3200+):

1. We found 15147 near-collision pairs by 500000 trials, which agrees with our theoretical estimate because $500000/15147 = 33.0098\cdots$.
2. From a single near-collision pair, we found 249630 full collision pairs by 500000 trials, which also agrees with our theoretical complexity because $500000/249630 \approx 2$.

[2] Our differential is used in an attack on 4-pass HAVAL by H.Yu et al. [7], but it is new for 3-pass HAVAL.

It took about one minute for the first 500000 trials. It also took about one minute for the next 500000 trials.

Related wroks:

- Modular differential attack was presented in 1997 by X.Y.Wang [3] and formalized in Eurocrypt '05 [5,6]. They showed that it is very powerful to break MD4, MD5, SHA-0, SHA-1 and HAVAL. Our attack is also based on the *modular differential* approach.
- On 4-pass HAVAL, H.Yu et al. [7] showed two two-block collision attacks that require 2^{43} and 2^{36} computations of 4-pass HAVAL, respectively. On 5-pass HAVAL, the H.Yu et al. [7] showed a one-block collision attack with 2^{123} computations of the compression function.

This paper is organized as follows. In Section 2, we provide a simple description of 3-pass HAVAL. In Section 3, we give an outline of our attack. In Section 4, we present the algorithm of our attack, and calculate the complexity. In Section 5, we report on our computational experiment and a collision example. In Section 6, we conclude this paper. In this paper, almost Tables and Figures are in the Appendix.

2 3-Pass HAVAL

HAVAL consists of three phases: (1) message padding phase, (2) main hashing phase and (3) optional compression phase.

2.1 Message Padding Phase

HAVAL pads an input message by appending some bit string so that its bit-length becomes a multiple of 1024.

2.2 Main Hashing Phase

HAVAL is a Merkle-Damgård hash function based on a compression function H as follows. Let $M_0||M_1||\cdots||M_t$ be the padded message, where $|M_i| = 1024$. Then for $i = 0, \cdots, t$, compute

$$IV_{i+1} = H(IV_i, M_i)$$

where $|IV_i| = 256$ and $IV_0 = (a, b, c, d, e, f, g, h)$ is the initial value such that

$$a = 0x243f6a88, b = 0x85a308d3,\ c = 0x13198a2e,\ d = 0x03707344,$$
$$e = 0xa4093822,\ f = 0x299f31d0, g = 0x082efa98, h = 0xec4e6c89.$$

The hashed value is given by IV_{t+1}.

H is described as follows. First define three functions as follows.

$$\begin{aligned}
F_1(x_0,x_1,x_2,x_3,x_4,x_5,x_6) &= (x_2 \bullet x_3) \oplus (x_2 \bullet x_4) \oplus x_4 \oplus (x_0 \bullet x_6) \oplus (x_1 \bullet x_5),\\
F_2(x_0,x_1,x_2,x_3,x_4,x_5,x_6) &= (x_0 \bullet x_2) \oplus (x_1 \bullet x_2) \oplus (x_1 \bullet x_3) \oplus (x_0 \bullet x_3 \bullet x_5)\\
&\quad \oplus (x_1 \bullet x_2 \bullet x_5) \oplus (x_3 \bullet x_5) \oplus (x_4 \bullet x_5)\\
&\quad \oplus x_6 \oplus (x_5 \bullet x_6),\\
F_3(x_0,x_1,x_2,x_3,x_4,x_5,x_6) &= x_0 \quad \oplus (x_0 \bullet x_3) \oplus (x_1 \bullet x_4) \oplus (x_2 \bullet x_5)\\
&\quad \oplus (x_3 \bullet x_4 \bullet x_5) \oplus (x_3 \bullet x_6),
\end{aligned}$$

where x_i is a 32-bit word, $x_i \bullet x_j$ is the bit-wise multiplication of x_i and x_j, and $x_i \oplus x_j$ is the bit-wise modulo 2 addition.

H next runs the following algorithm $\tilde{H}$ on input

$$\begin{aligned}
IV &= (a_0, b_0, c_0, d_0, e_0, f_0, g_0, h_0),\\
M &= (m_0, m_1, \ldots, m_{31}),
\end{aligned}$$

where each of $a_0, \ldots, h_0$ and m_i is a 32-bit word.

For($i = 0$ to 95){

$$\begin{aligned}
j &:= \lfloor i/32 \rfloor + 1;\\
p_i &:= F_j(a_i, b_i, c_i, d_i, e_i, f_i, g_i);\\
a_{i+1} &:= (p_i \ggg 7) + (h_i \ggg 11) + m_{ord(i)} + k_i \bmod 2^{32}; \qquad (1)\\
b_{i+1} &:= a_i,\ c_{i+1} := b_i,\ d_{i+1} := c_i,\ e_{i+1} := d_i,\\
f_{i+1} &:= e_i,\ g_{i+1} := f_i,\ h_{i+1} := g_i;
\end{aligned}$$

}

where $x \ggg s$ denotes the s-bit right rotation of x, $+$ denotes the modulo 2^{32} addition, and the word processing orders $ord(i)$ and the constant values k_i are given in Table 3. Note that $\tilde{H}$ consists of 96 rounds, the 0-th round through the 95-th round.

Finally, H outputs the following 256-bit value

$$\begin{aligned}
&IV + \tilde{H}(IV, M)\\
&= (a_0 + a_{96}, b_0 + b_{96}, c_0 + c_{96}, d_0 + d_{96}, e_0 + e_{96}, f_0 + f_{96}, g_0 + g_{96}, h_0 + h_{96}).
\end{aligned}$$

Figure 1 is an outline sketch of H. In the Appendix, we provide more detailed sketches for each round. (See Figures 2, 3, and 4.)

2.3 Optional Compression Phase

HAVAL supports hash-sizes of 128, 160, 192, 224 and 256 bits. The main algorithm computes 256-bit hash-values, and the other sizes are obtained by post-processing the 256-bit hash-value.

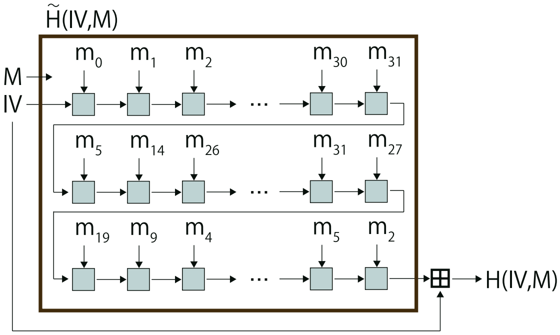


Fig. 1. The compression function H of 3-pass HAVAL

3 Outline of Our Attack

3.1 Notation

For $H(IV, M)$, $a_i, b_i, \cdots$ denote the local values which appear in the i-th round. Similarly, for $H(IV, M')$, $a'_i, b'_i, \cdots$ denote the local values which appear in the i-th round. We denote by $aa_0, bb_0, cc_0, dd_0, ee_0, ff_0, gg_0, hh_0$ the 8 words of IV_1. Define $\Delta a_i = a'_i - a_i \bmod 2^{32}$, and so on.

We denote by $x_{i,j}$ the j-th bit of 32-bit word x_i.

- $x'_i = x_i[j]$ means that x'_i is obtained by changing the jth bit of x_i from 0 to 1. That is, $x'_i = x_i$ except for that $x_{i,j} = 0$ and $x'_{i,j} = 1$.
- $x'_i = x_i[-j]$ means that x'_i is obtained by changing the jth bit of x_i from 1 to 0. That is, $x'_i = x_i$ except for that $x_{i,j} = 1$ and $x'_{i,j} = 0$.
- $x'_i = x_i[\pm j]$ means that $x_{i,j} \neq x'_{i,j}$.
- For example, $a_{14}[-21, 22]$ is the value obtained by modifying the 21-st and 22-nd bit of a_{14} from 1 to 0 and 0 to 1, respectively.

3.2 Attack

We show an efficient method to find a two-block (2048-bit) collision pair

$$M_0||M_1 = (m_{0,0}, \ldots, m_{0,31}, m_{1,0}, \ldots, m_{1,31})$$
$$M'_0||M'_1 = (m'_{0,0}, \ldots, m'_{0,31}, m'_{1,0}, \ldots, m'_{1,31}).$$

The proposed method first finds a near collision pair (M_0, M'_0) such that

$$\Delta IV_1 = IV'_1 - IV_1 = (0, 2^{31}, 0, 0, 0, 0, 0, 0)$$

We next find a pair (M_1, M'_1) such that

$$\begin{aligned}\Delta A &= \tilde{H}(IV'_1, M'_1) - \tilde{H}(IV_1, M_1) \\ &= (0, 2^{31}, 0, 0, 0, 0, 0, 0) \text{ or } (0, -2^{31}, 0, 0, 0, 0, 0, 0)\end{aligned} \tag{2}$$

Then it holds that

$$H(IV_1', M_1') = IV_1' + \tilde{H}(IV_1', M_1') = IV_1 + \tilde{H}(IV_1, M_1) = H(IV_1, M_1) \qquad (3)$$

That is,

$$\Delta IV_0 = 0 \overset{(M_0, M_0')}{\longrightarrow} \Delta IV_1 \overset{(M_1, M_1')}{\longrightarrow} \Delta A \rightarrow \Delta H = 0$$

Therefore, $(M_0||M_1, M_0'||M_1')$ is a collision pair.[3]

We use a message differential such that

$$\Delta m_{j,i} = m_{j,i}' - m_{j,i} \bmod 2^{32} = \begin{cases} 2^{31} & \text{if } i = 5, \\ 0 & \text{otherwise} \end{cases}$$

for $j = 0$ and 1.[4] That is,

$$\Delta M_0 = (0, 0, 0, 0, 0, 2^{31}, 0, \cdots, 0)$$

$$\Delta M_1 = (0, 0, 0, 0, 0, 2^{31}, 0, \cdots, 0)$$

Note that m_5 is the input to the 5-th, 32-nd, and 94-th round in each block because

$$5 = ord(5) = ord(32) = ord(94)$$

from Table 3. Now we will find the first block pair (M_0, M_0') that causes a local collision at the 32-nd round. Then (M_0, M_0') is automatically a near collision pair just after the 94-th round with difference $(0, \pm 2^{31}, 0, 0, 0, 0, 0, 0)$. This can be seen from the following table.

round	0	$\cdots$	5	$\cdots$	32	$\cdots$	94	95
Δm_i	$\Delta m_0 = 0$	$\cdots$	$\Delta m_5 = 2^{31}$	$\cdots$	Δm_5	$\cdots$	Δm_5	$\cdots$
$\tilde{H}(IV, M)$					collision		near collision	near collision

Similarly we will find the second block pair (M_1, M_1') which satisfies eq.(2), where $IV_1 = H(IV_0, M_0)$ and $IV_1' = H(IV_0, M_0')$. Then $M_0||M_0'$ and $M_1||M_1'$ are a full collision pair from eq.(3).

We present the (so called) *differential path* in Table 4 and Table 5.[5] In these Tables, for example,

round i	m_i'	Δa_{i+1}	Outputs $a_i', b_i', c_i', d_i', e_i', f_i', g_i', h_i'$
6	m_6	0	$a_7, a_6[32], a_5, a_4, a_3, a_2, a_1, a_0$

means that we want the outputs $(a_7', b_7', c_7', d_7', e_7', f_7', g_7', h_7')$ in the 6-th round of $H(IV', M')$ to be $(a_7, b_7[32], c_7, d_7, e_7, f_7, g_7, h_7)$. Note that $(a_7, b_7, c_7, d_7, e_7, f_7, g_7, h_7) = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$. We can find a full collision if all the conditions of these tables are satisfied.

[3] The operation + is the word-wise modular 2^{32} addition.

[4] This differential was used for 4-pass HAVAL [7]. Its complexity is 2^{43}.

[5] Table 4 was given in [7]. We constructed Table 5.

3.3 Sufficient Conditions

In Tables 6, 7, and 8, we present sufficient conditions for the differential path to hold. If Tables 6, 7, and 8 are satisfied, then Tables 4 and 5 are satisfied. As an example, we prove that the conditions for the 5-th and 6-th round given in Table 6 guarantee that the conditions for the 0-th to 6-th round shown in Table 4. The other conditions are derived similarly.

Since $\Delta m_0 = \cdots = \Delta m_4 = 0$, the differential path of the 0-th to the 4-th round holds. In the 5-th round of $H(IV_0', M_0')$, if the sufficient condition of the 5-th round in Table 6, that is, $a_{6,32} = 0$, then $a'_{6,32} = 1$ and $a'_{6,i} = a_{6,i}$ for $i \neq 32$ (i.e. $a'_6 = a_6[32]$), because

$$\begin{aligned}
a_6 &= (p_5 \gg 7) + (h_5 \gg 11) + m_{ord(5)} + k_5 \\
a'_6 &= (p'_5 \gg 7) + (h'_5 \gg 11) + m'_{ord(5)} + k_5 \\
&= (p_5 \gg 7) + (h_5 \gg 11) + (m_{ord(5)} + 2^{31}) + k_5 \\
&= (p_5 \gg 7) + (h_5 \gg 11) + m_{ord(5)} + k_5 + 2^{31} \\
&= a_6 + 2^{31}.
\end{aligned}$$

In the 6-th round of $H(IV_0', M_0')$, if the sufficient condition of the 5-th round in Table 6, that is, $a_{0,32} = 0$ then, since

$$\begin{aligned}
p_6 &= F_1(a_6, b_6, c_6, d_6, e_6, f_6, g_6) \\
&= F_1(a_6, a_5, a_4, a_3, a_2, a_1, a_0) \\
&= (a_4 \bullet a_3) \oplus (a_4 \bullet a_2) \oplus a_2 \oplus (a_6 \bullet a_0) \oplus (a_5 \bullet a_1),
\end{aligned}$$

we have

$$\begin{aligned}
p_{6,32} &= (a_{4,32} \bullet a_{3,32}) \oplus (a_{4,32} \bullet a_{2,32}) \oplus a_{2,32} \oplus (a_{6,32} \bullet a_{0,32}) \oplus (a_{5,32} \bullet a_{1,32}), \\
&= (a_{4,32} \bullet a_{3,32}) \oplus (a_{4,32} \bullet a_{2,32}) \oplus a_{2,32} \oplus (0 \bullet 0) \oplus (a_{5,32} \bullet a_{1,32}), \\
&= (a_{4,32} \bullet a_{3,32}) \oplus (a_{4,32} \bullet a_{2,32}) \oplus a_{2,32} \oplus 0 \oplus (a_{5,32} \bullet a_{1,32}), \\
p'_{6,32} &= (a'_{4,32} \bullet a'_{3,32}) \oplus (a'_{4,32} \bullet a'_{2,32}) \oplus a'_{2,32} \oplus (a'_{6,32} \bullet a'_{0,32}) \oplus (a'_{5,32} \bullet a'_{1,32}), \\
&= (a'_{4,32} \bullet a'_{3,32}) \oplus (a'_{4,32} \bullet a'_{2,32}) \oplus a'_{2,32} \oplus (1 \bullet 0) \oplus (a'_{5,32} \bullet a'_{1,32}), \\
&= (a'_{4,32} \bullet a'_{3,32}) \oplus (a'_{4,32} \bullet a'_{2,32}) \oplus a'_{2,32} \oplus 0 \oplus (a'_{5,32} \bullet a'_{1,32}),
\end{aligned}$$

Hence, $p'_6 = p_6$. Therefore $a'_7 = a_7$. The above equations for $p_{6,32}$ and $p'_{6,32}$ are clear by Figure 2.

4 Details

In this section, we present the details of our algorithm, and calculate the complexity to find a collision pair.

4.1 Our Algorithm

We observe that from $a_1, \cdots, a_{32}$, $M_0 = (m_0, \ldots, m_{31})$ is uniqluey determined from eq.(1), and $a_{33}, \cdots, a_{96}$ are also uniquely determined. Now in Table 6, all the rows except the last three rows specify the conditions on $a_1, \cdots, a_{32}$. Hence:

1. We choose $a_1, \cdots, a_{32}$ which satisfy these conditions randomly.
2. We compute $M_0 = (m_0, \ldots, m_{31})$ from eq.(1).[6]
3. If the last three rows are also satisfied, then we have done.

Next for given M_0, we apply the same strategy to find M_1.

1. We choose $a_1, \cdots, a_{32}$ which satisfy the conditions of Table 7 and Table 8 randomly.
2. We compute M_1 from eq.(1).
3. If the last row of Table 8 is also satisfied, then we have done.

```
Finding $M_0$
1. Randomly select $a_1, \ldots, a_{32}$
   that satisfy the sufficient conditions for the 0,1,2,...,31-th rounds.
2. For $i = 0$ to 31,
   $b_{i+1} := a_i$, $c_{i+1} := b_i$, $d_{i+1} := c_i$, $e_{i+1} := d_i$,
   $f_{i+1} := e_i$, $g_{i+1} := f_i$, $h_{i+1} := g_i$.
3. Calculate $p_0, \ldots, p_{31}$ of the algorithm of $H$ in Section 2
   and $m_0, \ldots, m_{31}$ as follows,
   $p_i := F_1(a_i, b_i, c_i, d_i, e_i, f_i, g_i)$,
   $m_i := a_{i+1} - (p_i \gg 7) - (h_i \gg 11) - k_i \bmod 2^{32}$.
4. Execute the 32-nd to the 95-th round of the compression function.
5. If $a_{95,32} = 1$, $a_{92,32} = 1$, $bb_{0,32} = 0$, $ff_{0,32} = 0$,
   $cc_{0,32} = dd_{0,32}$, and $aa_{0,32} = 0$,
   then fix $M_0 = (m_0, \ldots, m_{31})$.

Finding $M_1$
6.  Randomly select $a_1, \ldots, a_{32}$
    that satisfy the sufficient conditions for the 0,1,2,...,31-th rounds.
7.  For $i = 0$ to 31,
    $b_{i+1} := a_i$, $c_{i+1} := b_i$, $d_{i+1} := c_i$, $e_{i+1} := d_i$,
    $f_{i+1} := e_i$, $g_{i+1} := f_i$, $h_{i+1} := g_i$.
8.  Calculate $p_0, \ldots, p_{31}$ of the algorithm of $H$ in Section 2
    and $m_0, \ldots, m_{31}$ as follows,
    $p_i := F_1(a_i, b_i, c_i, d_i, e_i, f_i, g_i)$,
    $m_i := a_{i+1} - (p_i \gg 7) - (h_i \gg 11) - k_i \bmod 2^{32}$.
9.  Execute the 32-nd to the 95-th round of the compression function.
10. If $a_{92,32} = 1$, then output $(M_0||M_1)$ and $(M_0'||M_1')$ as a collision pair.
```

4.2 Success Probability

Assume that

$$Pr[x_{i,j} = 0] = Pr[x_{i,j} = 1] = 1/2 \tag{4}$$

[6] Note that our algorithm doesn't require message modification.

for any word x_i. We can find M_0 if the last three condition are satisfied in Table 6. Thereore, the success probability P of finding M_0 is given by

$$P = Pr[a_{95,32} = 1, a_{92,32} = 1, aa_{0,32} = 0, bb_{0,32} = 0, cc_{0,32} = dd_{0,32}, ff_{0,32} = 0,$$
$$= 1/2^5 \times Pr[bb_{0,32} = 0]$$

For bb_0, note that

$$bb_0 = b_0 + b_{96} = b + a_{95} = (10000101 \cdots) + (1??????? \cdots).$$

Thus, if $a_{95,31}$, $a_{95,30}$, $a_{95,29}$, or $a_{95,28}$ is 0, or $a_{95,31} = a_{95,30} = a_{95,29} = a_{95,28} = 1$ and $a_{95,27} = a_{95,26} = 0$, then $bb_{0,32} = 0$. Hence

$$\begin{aligned} & Pr[bb_{0,32} = 0] \\ \geq & Pr[a_{95,31} = 0, a_{95,30} = 0, a_{95,29} = 0, \text{ or } a_{95,28} = 0] \\ & + Pr[a_{95,31} = a_{95,30} = a_{95,29} = a_{95,28} = 1 \text{ and } a_{95,27} = a_{95,26} = 0] \\ = & (1 - Pr[a_{95,31} = 1, a_{95,30} = 1, a_{95,29} = 1, \text{ and } a_{95,28} = 1]) + 1/2^6 \\ = & (1 - 1/2^4) + 1/64 = 15/16 + 1/64 = 61/64. \end{aligned}$$

Therefore

$$P \geq 1/2^5 \times 61/64 = 61/2^{11} \approx 1/33.$$

Next suppose that the above M_0 is given. Then we can find M_1 if the last row of Table 8 is satisfied. Thereore, the success probability of finding M_1 is given by

$$Pr[a_{92,32} = 1] = 1/2.$$

4.3 How to Find Many Collisions

We can find many collision pairs from fixed (M_0, M_0') by running the algorithm "Finding M_1" many times. In this method, the complexity of finding k collision pairs is $2k + 33$.

5 Computational Experiment

We implemented our attack by a personal computer. First we found 15,147 desired M_0s by running the algorithm "Finding M_0" $500,000$ times. In this experiment, the success probability is $15,147/500,000 \approx 1/33$. It coincides with our theoretical probability shown in Section 3.

Next for fixed M_0, we found $249,630$ desired M_1s by running the algorithm "Finding M_1" $500,000$ times.[7] In this experiment, the success probability is $249,630/500,000 \approx 1/2$. It coincides with our theoretical probability shown in Section 3.

In total, we found $249,630$ full collision pairs by running the algorithms "Finding M_0" 39 times and "Finding M_1" $500,000$ times.

Consequently, our experiment supports our claim that we can find k collision pairs with $2k + 33$ computations of the compression functions. We illustrate one of the 249630 collision pairs in Table 2.

[7] It takes about one minute on our computer with CPU:AthlonXP 3200+.

Table 2. Collision example

M_0	c7f10962	08cf4e0c	ddfd60a8	597cbd0d	b050440c	205560d0	84569b2f	43b834dc
	1270d097	2b027ff7	32247646	8056892d	906feca6	a0a6b4ec	fbc11aca	d12586db
	f7e7bae1	ca89b85f	2d5a3e0f	8b4557da	8596d1bb	2bf5e1fd	b5b7f669	9445ea09
	343860ec	5c746759	bbce300c	d0985871	5229b382	8dab9e3e	f89f39d6	9179329b
M_1	cc4e7f72	c195d858	e5e2baf1	af7db590	84ddea8e	5990fd91	f6865ea5	9db928ce
	d3555dbd	6bf9b53a	694e5fff	e96766dc	2d541b98	d394d721	6a84b2c2	0d2bd1a1
	3afdac64	f0f67f58	60dd3e5d	aec84176	575012f1	24878a2f	304720ed	25eed9ae
	447f0e6e	b03eaa86	9fa12c2a	e98b9370	2e5cb01c	a2e23d56	cdaf12f2	2efb842d
M'_0	c7f10962	08cf4e0c	ddfd60a8	597cbd0d	b050440c	a05560d0	84569b2f	43b834dc
	1270d097	2b027ff7	32247646	8056892d	906feca6	a0a6b4ec	fbc11aca	d12586db
	f7e7bae1	ca89b85f	2d5a3e0f	8b4557da	8596d1bb	2bf5e1fd	b5b7f669	9445ea09
	343860ec	5c746759	bbce300c	d0985871	5229b382	8dab9e3e	f89f39d6	9179329b
M'_1	cc4e7f72	c195d858	e5e2baf1	af7db590	84ddea8e	d990fd91	f6865ea5	9db928ce
	d3555dbd	6bf9b53a	694e5fff	e96766dc	2d541b98	d394d721	6a84b2c2	0d2bd1a1
	3afdac64	f0f67f58	60dd3e5d	aec84176	575012f1	24878a2f	304720ed	25eed9ae
	447f0e6e	b03eaa86	9fa12c2a	e98b9370	2e5cb01c	a2e23d56	cdaf12f2	2efb842d
H	c9f26b47	513d34a2	0ad20a17	3d207470	04848b80	fc90cc0a	ef1cf172	d48c0d25

6 Conclusion

On 3-pass HAVAL, the best known attack finds a collision pair with 2^7 computations of the compression function. To find k collision pairs, it requires 2^7k computations.

In this paper, we presented a better collision attack on 3-pass HAVAL using modular differential method. It can find k collision pairs with only $2k + 33$ computations. Further, our message differential is different from the previous ones. (It is important to find collision pairs for different message differentials.)

References

1. Calyptix Security Corporation: HAVAL Version 1.1 (2003), http://labs.calyptix.com/haval.php
2. Van Rompay, B., Biryukov, A., Preneel, B., Vandewalle, J.: Cryptanalysis of 3-Pass HAVAL. In: Laih, C-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 228–245. Springer, Heidelberg (2003)
3. Wang, Xiaoyun: The Collision attack on SHA-0(in Chinese)(to appear,1997), http://www.infosec.sdu.edu.cn
4. Wang, X., Feng, D., Yu, X.: An attack on hash function HAVAL-128. Science in China Ser.F Information Sciences 48(5), 545–556 (2005)
5. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
6. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

7. Yu, H., Wang, X., Yun, A., Park, S.: Cryptanalysis of the Full HAVAL with 4 and 5 Passes. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 89–110. Springer, Heidelberg (2006)
8. Zheng, Y., Pieprzyk, J., Seberry, J.: HAVAL – A One-Way Hashing Algorithm with Variable Length of Output (Extended Abstract). In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 83–104. Springer, Heidelberg (1993)

Appendix: Tables and Figures

Table 3. Word processing orders $ord(i)$ and constant values k_i (hexadecimal numbers)

i	$ord(i)$
0 to 31	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 to 63	5 14 26 18 11 28 7 16 0 23 20 22 1 10 4 8 30 3 21 9 17 24 29 6 19 12 15 13 2 25 31 27
64 to 95	19 9 4 20 28 17 8 22 29 14 25 12 24 30 16 26 31 15 7 3 1 0 18 27 13 6 21 10 23 11 5 2

i	k_i
0 to 31	00000000
32 to 63	452821e6 38d01377 be5466cf 34e90c6c c0ac29b7 c97c50dd 3f84d5b5 b5470917 9216d5d9 8979fb1b d1310ba6 98dfb5ac 2ffd72db d01adfb7 b8e1afed 6a267e96 ba7c9045 f12c7f99 24a19947 b3916cf7 0801f2e2 858efc16 636920d8 71574e69 a458fea3 f4933d7e 0d95748f 728eb658 718bcd58 82154aee 7b54a41d c25a59b5
64 to 95	9c30d539 2af26013 c5d1b023 286085f0 ca417918 b8db38ef 8e79dcb0 603a180e 6c9e0e8b b01e8a3e d71577c1 bd314b27 78af2fda 55605c60 e65525f3 aa55ab94 57489862 63e81440 55ca396a 2aab10b6 b4cc5c34 1141e8ce a15486af 7c72e993 b3ee1411 636fbc2a 2ba9c55d 741831f6 ce5c3e16 9b87931e afd6ba33 6c24cf5c

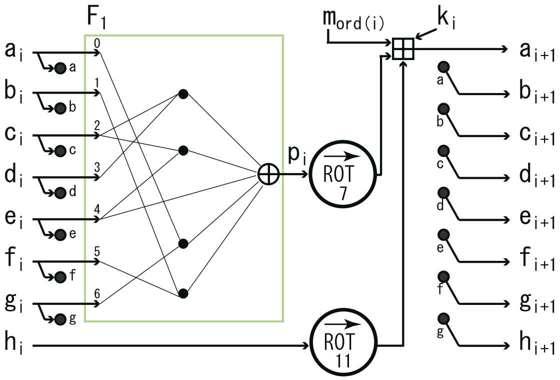


Fig. 2. The i-th round ($0 \le i \le 31$) in the compression function

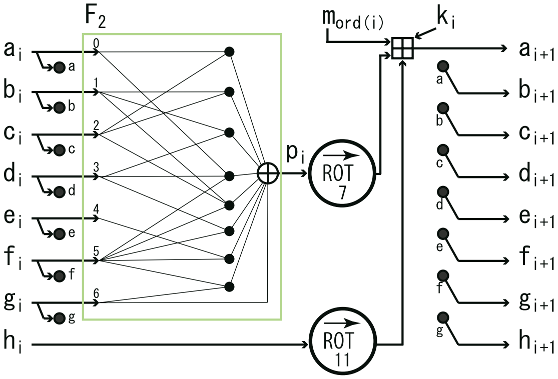


Fig. 3. The i-th round ($32 \le i \le 63$) in the compression function

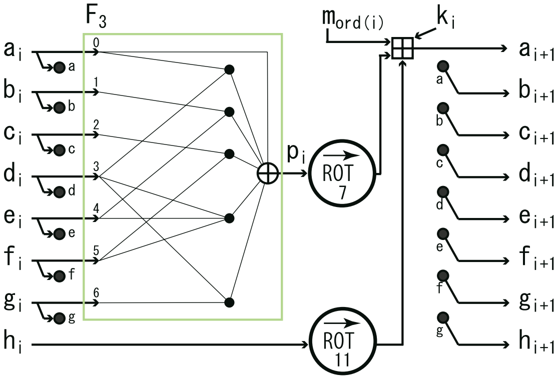


Fig. 4. The i-th round ($64 \le i \le 95$) in the compression function

Table 4. A differential path for $H(IV_0, M_0)$ and $H(IV_0, M'_0)$

round i	m'_i	Δa_{i+1}	Outputs $a'_i, b'_i, c'_i, d'_i, e'_i, f'_i, g'_i, h'_i$
IV'_0			$a_0, b_0, c_0, d_0, e_0, f_0, g_0, h_0$
0	m_0	0	$a_1, a_0, b_0, c_0, d_0, e_0, f_0, g_0$
1	m_1	0	$a_2, a_1, a_0, b_0, c_0, d_0, e_0, f_0$
2	m_2	0	$a_3, a_2, a_1, a_0, b_0, c_0, d_0, e_0$
3	m_3	0	$a_4, a_3, a_2, a_1, a_0, b_0, c_0, d_0$
4	m_4	0	$a_5, a_4, a_3, a_2, a_1, a_0, b_0, c_0$
5	m'_5	2^{31}	$a_6[32], a_5, a_4, a_3, a_2, a_1, a_0, b_0$
6	m_6	0	$a_7, a_6[32], a_5, a_4, a_3, a_2, a_1, a_0$
7	m_7	0	$a_8, a_7, a_6[32], a_5, a_4, a_3, a_2, a_1$
8	m_8	0	$a_9, a_8, a_7, a_6[32], a_5, a_4, a_3, a_2$
9	m_9	0	$a_{10}, a_9, a_8, a_7, a_6[32], a_5, a_4, a_3$
10	m_{10}	0	$a_{11}, a_{10}, a_9, a_8, a_7, a_6[32], a_5, a_4$
11	m_{11}	0	$a_{12}, a_{11}, a_{10}, a_9, a_8, a_7, a_6[32], a_5$
12	m_{12}	0	$a_{13}, a_{12}, a_{11}, a_{10}, a_9, a_8, a_7, a_6[32]$
13	m_{13}	2^{20}	$a_{14}[-21, 22], a_{13}, a_{12}, a_{11}, a_{10}, a_9, a_8, a_7$
14	m_{14}	0	$a_{15}, a_{14}[-21, 22], a_{13}, a_{12}, a_{11}, a_{10}, a_9, a_8$
15	m_{15}	0	$a_{16}, a_{15}, a_{14}[-21, 22], a_{13}, a_{12}, a_{11}, a_{10}, a_9$
16	m_{16}	0	$a_{17}, a_{16}, a_{15}, a_{14}[-21, 22], a_{13}, a_{12}, a_{11}, a_{10}$
17	m_{17}	-2^{14}	$a_{18}[15, 16, 17, -18], a_{17}, a_{16}, a_{15}, a_{14}[-21, 22], a_{13}, a_{12}, a_{11}$
18	m_{18}	0	$a_{19}, a_{18}[15, 16, 17, -18], a_{17}, a_{16}, a_{15}, a_{14}[-21, 22], a_{13}, a_{12}$
19	m_{19}	0	$a_{20}, a_{19}, a_{18}[15, 16, 17, -18], a_{17}, a_{16}, a_{15}, a_{14}[-21, 22], a_{13}$
20	m_{20}	0	$a_{21}, a_{20}, a_{19}, a_{18}[15, 16, 17, -18], a_{17}, a_{16}, a_{15}, a_{14}[-21, 22]$
21	m_{21}	0	$a_{22}, a_{21}, a_{20}, a_{19}, a_{18}[15, 16, 17, -18], a_{17}, a_{16}, a_{15}$
22	m_{22}	0	$a_{23}, a_{22}, a_{21}, a_{20}, a_{19}, a_{18}[15, 16, 17, -18], a_{17}, a_{16}$
23	m_{23}	0	$a_{24}, a_{23}, a_{22}, a_{21}, a_{20}, a_{19}, a_{18}[15, 16, 17, -18], a_{17}$
24	m_{24}	2^{10}	$a_{25}[11], a_{24}, a_{23}, a_{22}, a_{21}, a_{20}, a_{19}, a_{18}[15, 16, 17, -18]$
25	m_{25}	0	$a_{26}, a_{25}[11], a_{24}, a_{23}, a_{22}, a_{21}, a_{20}, a_{19}$
26	m_{26}	0	$a_{27}, a_{26}, a_{25}[11], a_{24}, a_{23}, a_{22}, a_{21}, a_{20}$
27	m_{27}	0	$a_{28}, a_{27}, a_{26}, a_{25}[11], a_{24}, a_{23}, a_{22}, a_{21}$
28	m_{28}	0	$a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[11], a_{24}, a_{23}, a_{22}$
29	m_{29}	0	$a_{30}, a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[11], a_{24}, a_{23}$
30	m_{30}	0	$a_{31}, a_{30}, a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[11], a_{24}$
31	m_{31}	0	$a_{32}, a_{31}, a_{30}, a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[11]$
32	m'_5	0	$a_{33}, a_{32}, a_{31}, a_{30}, a_{29}, a_{28}, a_{27}, a_{26}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
94	m'_5	-2^{31}	$a_{95}[-32], a_{94}, a_{93}, a_{92}, a_{91}, a_{90}, a_{89}, a_{88}$
95	m_2	0	$a_{96}, a_{95}[-32], a_{94}, a_{93}, a_{92}, a_{91}, a_{90}, a_{89}$
IV'_1			$aa_0, bb_0[32], cc_0, dd_0, ee_0, ff_0, gg_0, hh_0$

Table 5. A differential path for $H(IV_1, M_1)$ and $H(IV'_1, M'_1)$

round i	m'_i	Δa_{i+1}	Outputs $a'_i, b'_i, c'_i, d'_i, e'_i, f'_i, g'_i, h'_i$
IV'_1			$a_0, b_0[32], c_0, d_0, e_0, f_0, g_0, h_0$
0	m_0	0	$a_1, a_0, b_0[32], c_0, d_0, e_0, f_0, g_0$
1	m_1	0	$a_2, a_1, a_0, b_0[32], c_0, d_0, e_0, f_0$
2	m_2	0	$a_3, a_2, a_1, a_0, b_0[32], c_0, d_0, e_0$
3	m_3	0	$a_4, a_3, a_2, a_1, a_0, b_0[32], c_0, d_0$
4	m_4	0	$a_5, a_4, a_3, a_2, a_1, a_0, b_0[32], c_0$
5	m'_5	2^{31}	$a_6[32], a_5, a_4, a_3, a_2, a_1, a_0, b_0[32]$
6	m_6	2^{20}	$a_7[21], a_6[32], a_5, a_4, a_3, a_2, a_1, a_0$
7	m_7	0	$a_8, a_7[21], a_6[32], a_5, a_4, a_3, a_2, a_1$
8	m_8	-2^{24}	$a_9[25, 26, 27, -28], a_8, a_7[21], a_6[32], a_5, a_4, a_3, a_2$
9	m_9	0	$a_{10}, a_9[25, 26, 27, -28], a_8, a_7[21], a_6[32], a_5, a_4, a_3$
10	m_{10}	-2^{18}	$a_{11}[19, 20, -21], a_{10}, a_9[25, 26, 27, -28], a_8, a_7[21], a_6[32], a_5, a_4$
11	m_{11}	0	$a_{12}, a_{11}[19, 20, -21], a_{10}, a_9[25, 26, 27, -28], a_8, a_7[21], a_6[32], a_5$
12	m_{12}	-2^{11}	$a_{13}[12, 13, 14, 15, 16, -17], a_{12}, a_{11}[19, 20, -21], a_{10}, a_9[25, 26, 27, -28], a_8, a_7[21], a_6[32]$
13	m_{13}	-2^7	$a_{14}[-8], a_{13}[12, 13, 14, 15, 16, -17], a_{12}, a_{11}[19, 20, -21], a_{10}, a_9[25, 26, 27, -28], a_8, a_7[21]$
14	m_{14}	0	$a_{15}, a_{14}[-8], a_{13}[12, 13, 14, 15, 16, -17], a_{12}, a_{11}[19, 20, -21], a_{10}, a_9[25, 26, 27, -28], a_8$
15	m_{15}	0	$a_{16}, a_{15}, a_{14}[-8], a_{13}[12, 13, 14, 15, 16, -17], a_{12}, a_{11}[19, 20, -21], a_{10}, a_9[25, 26, 27, -28]$
16	m_{16}	0	$a_{17}, a_{16}, a_{15}, a_{14}[-8], a_{13}[12, 13, 14, 15, 16, -17], a_{12}, a_{11}[19, 20, -21], a_{10}$
17	m_{17}	0	$a_{18}, a_{17}, a_{16}, a_{15}, a_{14}[-8], a_{13}[12, 13, 14, 15, 16, -17], a_{12}, a_{11}[19, 20, -21]$
18	m_{18}	0	$a_{19}, a_{18}, a_{17}, a_{16}, a_{15}, a_{14}[-8], a_{13}[12, 13, 14, 15, 16, -17], a_{12}$
19	m_{19}	0	$a_{20}, a_{19}, a_{18}, a_{17}, a_{16}, a_{15}, a_{14}[-8], a_{13}[12, 13, 14, 15, 16, -17]$
20	m_{20}	0	$a_{21}, a_{20}, a_{19}, a_{18}, a_{17}, a_{16}, a_{15}, a_{14}[-8]$
21	m_{21}	-2^{28}	$a_{22}[-29], a_{21}, a_{20}, a_{19}, a_{18}, a_{17}, a_{16}, a_{15}$
22	m_{22}	-2^{21}	$a_{23}[22, 23, 24, -25], a_{22}[-29], a_{21}, a_{20}, a_{19}, a_{18}, a_{17}, a_{16}$
23	m_{23}	-2^{14}	$a_{24}[15, 16, 17, -18], a_{23}[22, 23, 24, -25], a_{22}[-29], a_{21}, a_{20}, a_{19}, a_{18}, a_{17}$
24	m_{24}	-2^{10}	$a_{25}[-11], a_{24}[15, 16, 17, -18], a_{23}[22, 23, 24, -25], a_{22}[-29], a_{21}, a_{20}, a_{19}, a_{18}$
25	m_{25}	0	$a_{26}, a_{25}[-11], a_{24}[15, 16, 17, -18], a_{23}[22, 23, 24, -25], a_{22}[-29], a_{21}, a_{20}, a_{19}$
26	m_{26}	0	$a_{27}, a_{26}, a_{25}[-11], a_{24}[15, 16, 17, -18], a_{23}[22, 23, 24, -25], a_{22}[-29], a_{21}, a_{20}$
27	m_{27}	0	$a_{28}, a_{27}, a_{26}, a_{25}[-11], a_{24}[15, 16, 17, -18], a_{23}[22, 23, 24, -25], a_{22}[-29], a_{21}$
28	m_{28}	0	$a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[-11], a_{24}[15, 16, 17, -18], a_{23}[22, 23, 24, -25], a_{22}[-29]$
29	m_{29}	0	$a_{30}, a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[-11], a_{24}[15, 16, 17, -18], a_{23}[22, 23, 24, -25]$
30	m_{30}	0	$a_{31}, a_{30}, a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[-11], a_{24}[15, 16, 17, -18]$
31	m_{31}	0	$a_{32}, a_{31}, a_{30}, a_{29}, a_{28}, a_{27}, a_{26}, a_{25}[-11]$
32	m'_5	0	$a_{33}, a_{32}, a_{31}, a_{30}, a_{29}, a_{28}, a_{27}, a_{26}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
94	m'_5	$\pm 2^{31}$	$a_{95}[\pm 32], a_{94}, a_{93}, a_{92}, a_{91}, a_{90}, a_{89}, a_{88}$
95	m_2	0	$a_{96}, a_{95}[\pm 32], a_{94}, a_{93}, a_{92}, a_{91}, a_{90}, a_{89}$
IV'_2			$a_0+a_{96}, b_0[32]+a_{95}[\pm 32], c_0+a_{94}, d_0+a_{93}, e_0+a_{92}, f_0+a_{91}, g_0+a_{90}, h_0+a_{89}$ Full Collision !!

Table 6. Sufficient conditions on a_i for the differential path in Table 4

round i	Sufficient conditions for each round
$5-12$	$a_{6,32}=0,\ a_{0,32}=0,\ a_{2,32}=0,\ a_{5,32}=a_{4,32},\ a_{7,32}=0,\ a_{8,32}=1,\ a_{10,32}=0,\ a_{12,32}=0$
13	$a_{14,21}=1, a_{14,22}=0$
14	$a_{8,21}=0, a_{8,22}=0$
15	$a_{10,21}=0, a_{10,22}=0$
16	$a_{13,21}=a_{12,21}, a_{13,22}=a_{12,22}$
17	$a_{18,15}=0,\ a_{18,16}=0,\ a_{18,17}=0,\ a_{18,18}=1,\ a_{15,21}=0,\ a_{15,22}=1,\ a_{11,22}=0, a_{12,22}=1, a_{13,22}=1, a_{16,22}=1$
18	$a_{12,15}=0, a_{12,16}=0, a_{12,17}=0, a_{12,18}=0, a_{16,21}=1, a_{16,22}=1$
19	$a_{14,15}=0, a_{14,16}=0, a_{14,17}=0, a_{14,18}=0, a_{18,21}=0, a_{18,22}=0$
20	$a_{16,15}=a_{17,15}, a_{16,16}=a_{17,16},\ a_{16,17}=a_{17,17},\ a_{16,18}=a_{17,18},\ a_{20,21}=0, a_{20,22}=0$
21	$a_{19,15}=0, a_{19,16}=0, a_{19,17}=1, a_{19,18}=0, a_{15,17}=1, a_{16,17}=0, a_{17,17}=0, a_{21,17}=1$
22	$a_{20,15}=1, a_{20,16}=1, a_{20,17}=1, a_{20,18}=1$
23	$a_{22,15}=0, a_{22,16}=0, a_{22,17}=0, a_{22,18}=0$
24	$a_{24,15}=0, a_{24,16}=0, a_{24,17}=0, a_{24,18}=1, a_{25,11}=0$
25	$a_{19,11}=1, a_{20,11}=0, a_{22,11}=0, a_{21,11}=0$
$26-31$	$a_{21,11}=0,\ a_{24,11}=a_{23,11},\ a_{26,11}=0,\ a_{27,11}=1,\ a_{29,11}=0,\ a_{31,11}=0$
$94-95$	$a_{95,32}=1,\ a_{92,32}=1$
IV_1	$aa_{0,32}=0, bb_{0,32}=0, cc_{0,32}=dd_{0,32}, ff_{0,32}=0$

Table 7. Sufficient conditions on a_i for the 0,...,11-th rounds of the differential path in Table 5

round i	Sufficient conditions for each round
0	$f_{0,32}=0$
1	$c_{0,32}=d_{0,32}$
2	$a_{0,32}=0$
3	$a_{1,32}=1$
4	$a_{3,32}=0$
5	$a_{5,32}=0, a_{6,32}=0$
6	$a_{7,21}=0, a_{0,32}=0$
7	$a_{1,21}=0, a_{2,32}=0$
8	$a_{9,25}=0, a_{9,26}=0, a_{9,27}=0, a_{9,28}=1, a_{3,21}=0, a_{2,32}=0, a_{3,32}=0,\ a_{4,32}=1, a_{5,32}=0$
9	$a_{7,32}=0, a_{5,21}=a_{6,21}, a_{3,25}=0, a_{3,26}=0, a_{3,27}=0, a_{3,28}=0$
10	$a_{11,19}=0, a_{11,20}=0, a_{11,21}=1, a_{8,32}=1, a_{8,21}=0, a_{5,25}=0, a_{5,26}=1, a_{5,27}=0, a_{5,28}=0, a_{8,26}=0, a_{6,26}=1, a_{4,26}=0$
11	$a_{5,19}=0, a_{5,20}=0, a_{9,21}=a_{5,21}+1, a_{7,25}=a_{8,25}, a_{7,26}=a_{8,26}, a_{8,27}=a_{7,27}, a_{7,28}=a_{8,28}, a_{10,32}=0$

Table 8. Sufficient conditions on a_i for the 12,. . .,95-th rounds of the differential path in Table 5

round i	Sufficient conditions for each round
12	$a_{13,12}=0, a_{13,13}=0, a_{13,14}=0, a_{13,15}=0, a_{13,16}=0, a_{13,17}=1, a_{12,32}=0, a_{7,19}=1, a_{7,20}=0, a_{10,25}=0, a_{10,26}=0, a_{10,27}=0, a_{10,28}=0, a_{8,19}=1, a_{10,19}=0, a_{6,19}=0$
13	$a_{14,8}=1, a_{7,12}=0, a_{7,13}=0, a_{7,14}=0, a_{7,15}=1, a_{7,16}=0, a_{7,17}=0, a_{10,19}=a_{9,19}, a_{10,20}=a_{9,20}, a_{10,21}=a_{9,21}+1, a_{11,25}=1, a_{11,26}=1, a_{11,27}=1, a_{11,28}=0, a_{7,28}=0, a_{8,28}=0, a_{8,15}=0, a_{9,15}=0, a_{10,15}=1, a_{11,15}=1, a_{13,21}=1$
14	$a_{8,8}=0, a_{9,12}=0, a_{9,13}=0, a_{9,14}=0, a_{9,15}=0, a_{9,16}=0, a_{9,17}=1, a_{12,19}=0, a_{12,20}=0, a_{12,21}=0, a_{13,25}=0, a_{13,26}=0, a_{13,27}=0, a_{13,28}=0, a_{8,17}=0, a_{10,17}=0, a_{11,17}=0$
15	$a_{10,8}=0, a_{12,12}=a_{11,12}, a_{12,13}=a_{11,13}, a_{12,14}=a_{11,14}, a_{12,15}=a_{11,15}, a_{12,16}=a_{11,16}, a_{12,17}=a_{11,17}, a_{13,19}=1, a_{13,20}=1, a_{13,21}=1, a_{15,25}=0, a_{15,26}=0, a_{15,27}=0, a_{15,28}=0$
16	$a_{13,8}=a_{12,8}, a_{14,12}=0, a_{14,13}=0, a_{14,14}=0, a_{14,15}=0, a_{14,16}=0, a_{14,17}=0, a_{15,19}=0, a_{15,20}=0, a_{15,21}=1, a_{10,21}=0, a_{12,21}=0, a_{13,21}=1, a_{14,21}=1$
17	$a_{15,8}=0, a_{15,12}=1, a_{15,13}=1, a_{15,14}=1, a_{15,15}=1, a_{15,16}=1, a_{15,17}=1, a_{17,19}=0, a_{17,20}=0, a_{17,21}=0$
18	$a_{16,8}=1, a_{17,12}=0, a_{17,13}=0, a_{17,14}=0, a_{17,15}=1, a_{17,16}=0, a_{17,17}=0, a_{14,15}=0, a_{16,15}=0, a_{18,15}=0$
19	$a_{18,8}=0, a_{19,12}=0, a_{19,13}=0, a_{19,14}=0, a_{19,15}=0, a_{19,16}=0, a_{19,17}=0$
20	$a_{15,8}=0, a_{16,8}=1, a_{18,8}=0, a_{20,8}=1$
21	$a_{22,29}=1$
22	$a_{23,22}=0,\ a_{23,23}=0,\ a_{23,24}=0,\ a_{23,25}=1,\ a_{16,19}=1,\ a_{17,29}=0,\ a_{18,29}=0,\ a_{19,29}=0$
23	$a_{24,15}=0, a_{24,16}=0, a_{24,17}=0, a_{24,18}=1, a_{18,29}=0, a_{17,22}=1, a_{17,23}=0, a_{17,24}=0, a_{17,25}=0, a_{18,22}=0, a_{19,22}=0, a_{20,22}=1, a_{21,22}=1$
24	$a_{25,11}=1, a_{21,29}=a_{20,29}, a_{19,22}=0, a_{19,23}=0, a_{19,24}=0, a_{19,25}=0, a_{18,15}=0, a_{18,16}=0, a_{18,17}=0, a_{18,18}=1, a_{19,18}=0, a_{20,18}=0, a_{22,18}=0$
25	$a_{19,11}=0, a_{20,15}=0, a_{20,16}=0, a_{20,17}=0, a_{20,18}=0, a_{22,22}=a_{21,22}, a_{22,23}=a_{21,23}, a_{22,24}=a_{21,24}, a_{22,25}=a_{21,25}, a_{23,29}=0$
26	$a_{21,11}=0,\ a_{23,15}=a_{22,15},\ a_{23,16}=a_{22,16}, a_{23,17}=a_{22,17}, a_{23,18}=a_{22,18}, a_{24,22}=0, a_{24,23}=0, a_{24,24}=0, a_{24,25}=0, a_{24,29}=1,$
27	$a_{24,11}=a_{23,11}, a_{25,15}=0, a_{25,16}=0, a_{25,17}=0, a_{25,18}=0, a_{25,22}=1, a_{25,23}=1, a_{25,24}=1, a_{25,25}=1, a_{26,29}=0$
28	$a_{26,11}=0, a_{26,15}=1, a_{26,16}=1, a_{26,17}=1, a_{26,18}=1, a_{27,22}=0, a_{27,23}=0, a_{27,24}=0, a_{27,25}=0, a_{28,29}=0$
29	$a_{27,11}=1, a_{28,15}=0, a_{28,16}=0, a_{28,17}=0, a_{28,18}=0, a_{29,22}=0, a_{29,23}=0, a_{29,24}=0, a_{24,25}=0, a_{25,25}=1, a_{27,25}=0, a_{29,25}=1$
30	$a_{29,11}=0, a_{30,15}=0, a_{30,16}=0, a_{30,17}=0, a_{30,18}=1, a_{25,18}=0, a_{26,18}=1, a_{28,18}=0$
31	$a_{31,11}=1, a_{26,11}=0, a_{27,11}=1, a_{29,11}=0$
95	$a_{92,32}=1$

A Secure Threshold Anonymous Password-Authenticated Key Exchange Protocol

SeongHan Shin[1], Kazukuni Kobara[1], and Hideki Imai[1,2]

[1] Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST),
1-18-13, Sotokannda, Chiyoda-ku, Tokyo 101-0021 Japan
{seonghan.shin, k-kobara, h-imai}@aist.go.jp
[2] Chuo University,
1-13-27, Haruhi, Bunkyo-ku, Tokyo 112-8551 Japan

Abstract. At Indocrypt 2005, Viet et al., [20] have proposed an anonymous password-authenticated key exchange (PAKE) protocol and its threshold construction both of which are designed for client's password-based authentication and anonymity against a passive server, who does not deviate the protocol. In this paper, we first point out that their threshold construction is completely insecure against off-line dictionary attacks. For the threshold $t > 1$, we propose a secure threshold anonymous PAKE (for short, TAP) protocol with the number of clients n upper-bounded, such that $n \leq 2\sqrt{N-1} - 1$, where N is a dictionary size of passwords. We also show that the TAP protocol provides semantic security of session keys in the random oracle model, with the reduction to the computational Diffie-Hellman problem, as well as anonymity against a passive server. For the threshold $t = 1$, we propose an efficient anonymous PAKE protocol that significantly improves efficiency in terms of computation costs and communication bandwidth compared to the original (not threshold) anonymous PAKE protocol [20].

1 Introduction

In 1976, Diffie and Hellman published their seminal paper that introduced how to share a secret over public networks [9]. Since then, many researchers have tried to design secure cryptographic protocols for realizing secure channels. These protocols are necessary because application-oriented protocols are frequently developed assuming the existence of such secure channels. In the 2-party setting (e.g., a client and a server), this can be achieved by an authenticated key exchange (AKE) protocol at the end of which the two parties authenticate each other and share a common and temporal session key to be used for subsequent cryptographic algorithms (e.g., AES-CBC or MAC). For authentication, the parties typically share some information in advance. The shared information may be the form of high-entropy cryptographic keys: either a secret key that can be used for symmetric-key encryption or message authentication

A. Miyaji, H. Kikuchi, and K. Rannenberg (Eds.): IWSEC 2007, LNCS 4752, pp. 444–458, 2007.
© Springer-Verlag Berlin Heidelberg 2007

code (e.g., [7,16]), or public keys (while the corresponding private keys are kept secret) which can be used for public-key encryption or digital signatures (e.g., [10,22,2,16,12]).

In practice, low-entropy human-memorable passwords such as 4-digit pin-code or alphanumerical passwords are commonly used rather than high-entropy keys because of its convenience in use. Many password-based AKE protocols have been extensively investigated for a long time where a client remembers a short password and the corresponding server holds the password or its verification data that is used to verify the client's knowledge of the password. However, one should be careful about two major attacks on passwords: on-line and off-line dictionary attacks. The on-line dictionary attack is a series of exhaustive searches for a secret performed on-line, so that an adversary can sieve out possible secret candidates one by one communicating with the target party. In contrast, the off-line dictionary attack is performed off-line in parallel where an adversary exhaustively enumerates all possible secret candidates, in an attempt to determine the correct one, by simply guessing a secret and verifying the guessed secret with recorded transcripts of a protocol. While on-line attacks are applicable to all of the password-based protocols equally, they can be prevented by letting a server take appropriate intervals between invalid trials. But, we cannot avoid off-line attacks by such policies, mainly because the attacks can be performed off-line and independently of the party.

1.1 Password-Authenticated Key Exchange and Anonymity

In 1992, Bellovin and Merritt [4] discussed an interesting problem about how to design a secure password-only protocol where a client remembers his/her password only (without any device and any additional assumption) and the counterpart server has password verification data. Their proposed protocols are good examples (though some are turned out insecure) that a combination of symmetric and asymmetric cryptographic techniques can prevent an adversary from verifying a guessed password (i.e., doing off-line dictionary attacks). Later, their AKE protocols have formed the basis for what we call Password-Authenticated Key Exchange (PAKE) protocols. Such protocols have been in standardization of IEEE P1363.2 [11].

In PAKE protocols, a client should send his/her identity clearly in order to authenticate each other and share a master-secret that may be the Diffie-Hellman key or a shared secret to be used for generating authenticators and session keys. Let us suppose an adversary who fully controls the networks. Though the adversary cannot impersonate any party in PAKE protocols with non-negligible probability, it is easy to collect a client's personal information about the communication history itself (e.g., history of access to ftp servers, web-mail servers, Internet banking servers or shopping mall servers). These information may reflect the client's life pattern and sometimes can be used for spam mails. For this problem, Viet et al., [20] have proposed an anonymous PAKE protocol and its threshold construction[1] that simply combine a PAKE protocol [1] for generating secure channels with an Oblivious Transfer (OT) protocol [19,8] for client's anonymity. The anonymity is guaranteed against an outside adversary as well as a passive server, who follows the protocol honestly but it is curious about identity of client involved with the protocol. They also gave an application for a company's public bulletin board to which any employee can upload opinions in a password-authenticated and anonymous way. As discussed in [20], their (not threshold) anonymous PAKE protocol can not provide anonymity against an active server, who deviates the protocol by changing messages at its own (see Section 5 of [20]). Though they did not mention anything about their threshold construction, it may prevent an active server from obtaining information on the client's identity since any client can blend him/herself to the subgroup.

1.2 Our Contributions

Partly motivated from Nguyen's insights [14] on the relationship between PAKE protocols and other cryptographic primitives, we carefully revisit Viet et al's anonymous PAKE protocols [20]. In this paper, we first point out that Viet et al's threshold anonymous PAKE protocol is insecure against off-line dictionary attacks. For the threshold $t > 1$, we propose a secure threshold anonymous PAKE (for short, TAP) protocol that provides not only semantic security of session keys in the random oracle model with the reduction to the computational Diffie-Hellman problem but also anonymity against a passive server, who does not deviate the protocol but is curious about the clients' identities. We also give the condition on the number of clients n, such that $n \leq 2\sqrt{N-1} - 1$, for the optimal security result against on-line dictionary attacks where N is a dictionary size of passwords. For the threshold $t = 1$, we propose an efficient anonymous PAKE protocol that can be easily obtained from the TAP protocol. The resultant protocol significantly improves efficiency in terms of computation costs and communication bandwidth compared to the original (not threshold) anonymous PAKE protocol [20].

1.3 Organization

This paper is organized as follows. In the next section, we show that the previous threshold anonymous PAKE protocol is insecure against off-line attacks. Section 3 is assigned to security model. In Section 4, we propose a secure threshold anonymous PAKE (TAP) protocol. Section 5 is devoted to its security proofs, followed by discussion about the condition on n in Section 6. For the threshold $t = 1$, we also propose an efficient anonymous PAKE protocol in Section 7. Finally, we conclude in Section 8.

[1] Here, the "threshold" number of clients are involved with the protocol. In a different context, MacKenzie et al., [13] proposed a threshold PAKE protocol where the "threshold" number of servers collaborates one another to resist against compromise of the password verification data. However, the collaborations in the former/latter protocols assume secure channels among clients/servers, respectively.

2 The Previous Threshold Anonymous PAKE Protocol

In this section, we first give some notation to be used throughout this paper. Then we explain how the previous threshold anonymous PAKE protocol [20,21] works and show its insecurity against off-line dictionary attacks.

2.1 Notation

Let $\mathbb{G}_p$ be a finite, cyclic group of prime order p and g be a generator of $\mathbb{G}_p$, whose elements are quadratic residues modulo p. Let h be another generator of $\mathbb{G}_p$ so that its discrete logarithm problem with g (i.e., computing $b = \log_g h$) should be hard. The parameter $(\mathbb{G}_p, p, g, h)$ is given as public information. In the aftermath, all the subsequent arithmetic operations are performed in modulo p unless otherwise stated.

Let l denote the security parameter for hash functions. Let N be a dictionary size of passwords. Let $\{0,1\}^*$ denote the set of finite binary strings and $\{0,1\}^l$ the set of binary strings of length l. If D is a set, then $d \overset{R}{\leftarrow} D$ indicates the process of selecting d at random and uniformly over D. Let "$||$" denote the concatenation of bit strings in $\{0,1\}^\star$. Let "$\bigoplus$" denote the exclusive-OR (XOR) operation of bit strings. The hash functions $\mathcal{F}$ and $\mathcal{F}'$ are full-domain hash (FDH) functions, mapping $\{0,1\}^\star$ to $\mathbb{Z}_p^\star$. While $\mathcal{G} : \{0,1\}^\star \rightarrow \mathbb{G}_p$ is another FDH function, the others are denoted $\mathcal{H}_k : \{0,1\}^\star \rightarrow \{0,1\}^l$, for $k = 1, 2$ and 3, where $\mathcal{G}$ and $\mathcal{H}_k$ are distinct secure one-way hash functions. Let C and S be the identities of a set of all clients and server, respectively, with each ID $\in \{0,1\}^\star$.

2.2 Protocol Description

Here we describe the threshold anonymous PAKE (TA-PAKE) protocol [20,21] where any subgroup SG, consisting of at least t ($t \leq n$) clients among n clients, generates a session key with server S in a password-authenticated and anonymous way.[2] We assume that each client in the subgroup are connected via secure channels. See Fig. 1 for graphical description of the TA-PAKE protocol.

Step 1

1.1 By collaborating with one another, the subgroup SG chooses a random number x from $\mathbb{Z}_p^\star$ and computes $X \equiv g^x$.

1.2 Each client C_i ($1 \leq i \leq t$) chooses two random numbers $(r_i, s_i) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2$, and then computes $w_i \equiv h^{r_i \cdot \mathcal{F}(i, pw_{C_i})}$ and $A_i \equiv w_i \times g^{s_i}$ where i and pw_{C_i} are the index and the password, respectively, for client C_i. The r_i and s_i are kept secret by C_i.

[2] The only difference of [21] from [20] is that the subgroup SG chooses $w_l \overset{R}{\leftarrow} \mathbb{G}_p$, for $t+1 \leq l \leq n$, and sends $\{w_j\}_{1 \leq j \leq n}$ along with other values in the first flow. In fact, the TA-PAKE protocol of [20] doesn't work correctly since server S has no idea on w_j.

Public information: $(\mathbb{G}_p, p, g, h), \mathcal{F}, \mathcal{F}', \mathcal{G}, \mathcal{H}_1, \mathcal{H}_2, C = \{C_1, \cdots, C_n\}$

Subgroup $SG = \{C_1, \cdots, C_t\}$

$x \overset{R}{\leftarrow} \mathbb{Z}_p^\star, X \equiv g^x$

For each C_i $(1 \leq i \leq t)$,

$(r_i, s_i) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2$,

$w_i \equiv h^{r_i \cdot \mathcal{F}(i, pw_{C_i})}$,

and $A_i \equiv w_i \times g^{s_i}$.

For $l = t+1$ to n, $w_l \overset{R}{\leftarrow} \mathbb{G}_p$.

$$\xrightarrow{C, X, \{A_i\}_{1 \leq i \leq t}, \{w_j\}_{1 \leq j \leq n}}$$

Server S $\left(pw_{C_j},\ 1 \leq j \leq n\right)$

$(y, z) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2, Y \equiv g^y, Z \equiv g^z$

$f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k$ where

$u_0 = y$ and $u_k \overset{R}{\leftarrow} \mathbb{Z}_p^\star$

For $j = 1$ to n,

$y_j \leftarrow f(j)$,

$Y_j \equiv g^{y_j}$,

and $\alpha_j \leftarrow \mathcal{G}\left(w_j^z\right) \bigoplus \left(Y_j \times g^{\mathcal{F}'(pw_{C_j})}\right)$.

For $i = 1$ to t, $D_i \equiv A_i^z$.

$K_S \equiv X^y$

$V_S \leftarrow \mathcal{H}_1(T||Y||K_S)$

$$\xleftarrow{S, Z, \{D_i\}_{1 \leq i \leq t}, \{\alpha_j\}_{1 \leq j \leq n}, V_S}$$

$SK \leftarrow \mathcal{H}_2(T||Y||K_S)$

For each C_i $(1 \leq i \leq t)$,

$Y_i \equiv (\alpha_i \bigoplus \mathcal{G}(D_i / Z^{s_i})) \times \left(g^{\mathcal{F}'(pw_{C_i})}\right)^{-1}$.

$Y \equiv \prod_{i=1}^t Y_i^{\lambda_i}$ where $\lambda_i \equiv \prod_{1 \leq k \leq t, k \neq i} \frac{k}{k-i}$,

and $K_C \equiv Y^x$

If $V_S \neq \mathcal{H}_1(T||Y||K_C)$, reject.

Otherwise, $SK \leftarrow \mathcal{H}_2(T||Y||K_C)$

and accept.

Fig. 1. The threshold anonymous PAKE (TA-PAKE) protocol [20,21] where $T = C||S||X||Z||\{D_i\}_{1 \leq i \leq t}||\{\alpha_j\}_{1 \leq j \leq n}$

1.3 The subgroup SG chooses $n-t$ random numbers w_l from $\mathbb{G}_p$, and then sends $C, X, \{A_i\}_{1 \leq i \leq t}$ and $\{w_j\}_{1 \leq j \leq n}$ to server S.

Step 2

2.1 The server chooses two random numbers $(y, z) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2$ and computes $(Y \equiv g^y, Z \equiv g^z)$ where the exponent y is distributed as shares by using Shamir's secret sharing scheme [15]. Specifically, server S generates the respective share $f(j)$, for n clients, from a random polynomial $f(x)$ of degree $t-1$ with coefficients u_k $(1 \leq k \leq t-1)$ in $\mathbb{Z}_p^\star$

$$f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k \tag{1}$$

and sets $u_0 = y$.

2.2 For j $(1 \leq j \leq n)$, server S computes $Y_j \equiv g^{f(j)}$ and $\alpha_j \leftarrow \mathcal{G}\left(w_j^z\right) \bigoplus \left(Y_j \times g^{\mathcal{F}'(pw_{C_j})}\right)$.

2.3 For i $(1 \leq i \leq t)$, server S computes $D_i \equiv A_i^z$.

2.4 The server computes $K_S \equiv X^y$, from which its authenticator V_S and session key SK are derived as follows: $V_S \leftarrow \mathcal{H}_1(C||S||X||Z||\{D_i\}_{1\leq i\leq t}|| \{\alpha_j\}_{1\leq j\leq n}||Y||K_S)$ and $SK \leftarrow \mathcal{H}_2(C||S||X||Z||\{D_i\}\ _{1\leq i\leq t}||\{\alpha_j\}_{1\leq j\leq n}|| Y||K_S)$. Then, server S sends $S, Z, \{D_i\}_{1\leq i\leq t}, \{\alpha_j\}_{1\leq j\leq n}$ and V_S to subgroup SG.

Step 3

3.1 Each client C_i $(1 \leq i \leq t)$ extracts $Y_i \equiv (\alpha_i \bigoplus \mathcal{G}(D_i/Z^{s_i})) \times \left(g^{\mathcal{F}'(pw_{C_i})}\right)^{-1}$.

3.2 By collaborating with one another, subgroup SG recovers Y from Y_i and computes $K_C \equiv Y^x$. Note that the Y can be reconstructed from the shares of any qualified subgroup of clients by Lagrange interpolation.

3.3 If V_S is valid, subgroup SG computes a session key SK as follows: $SK \leftarrow \mathcal{H}_2(C||S||X||Z||\{D_i\}\ _{1\leq i\leq t}||\{\alpha_j\}_{1\leq j\leq n}||Y||K_C)$. Otherwise, it terminates.

2.3 Insecurity of TA-PAKE Protocol

We show that the TA-PAKE protocol [20,21] is insecure against off-line dictionary attacks. First we suppose that an adversary $\mathcal{A}$ impersonates the subgroup SG without knowing any password.

Step 1'

1.1 An adversary $\mathcal{A}$ chooses a random number x from $\mathbb{Z}_p^\star$ and computes $X \equiv g^x$, and also chooses n random numbers $w_j \overset{R}{\leftarrow} \mathbb{G}_p$, for $1 \leq j \leq n$.

1.2 For each client C_i $(1 \leq i \leq t)$, adversary $\mathcal{A}$ chooses a random number $s_i \overset{R}{\leftarrow} \mathbb{Z}_q^\star$ and then computes $A_i \equiv w_i \times g^{s_i}$. The adversary sends $C, X, \{A_i\}_{1\leq i\leq t}$ and $\{w_j\}_{1\leq j\leq n}$ to server S.

Step 3'

3.1 After receiving the message from server S, adversary $\mathcal{A}$ performs the following: compute Y_i', as the honest client C_i of subgroup SG would do, with all of the possible password candidates pw'_{C_i} and store N different Y_i', for each client C_i $(1 \leq i \leq t)$.

$$Y_i' \equiv \left(\alpha_i \bigoplus \mathcal{G}(D_i/Z^{s_i})\right) \times \left(g^{\mathcal{F}'(pw'_{C_i})}\right)^{-1} \tag{2}$$

.

3.2 With tN different Y_i', the adversary recovers $Y' \equiv \prod_{i=1}^{t} Y_i'^{\lambda_i}$ and the latter is used to compute $K_C' \equiv Y'^x$. Finally, adversary $\mathcal{A}$ can find out the correct $\{pw'_{C_1}, \cdots, pw'_{C_t}\}$ by checking whether a subgroup of password candidates satisfies $V_S = \mathcal{H}_1(C||S||X||Z||\{D_i\}_{1\leq i\leq t}||\{\alpha_j\}_{1\leq j\leq n}||Y'||K_C')$ or not. Note that each subgroup guarantees a unique polynomial $f'(x)$ of degree $t-1$.

In the worst case, adversary $\mathcal{A}$ can find out $\{pw'_{C_1}, \cdots, pw'_{C_t}\}$ after N^t trials. Though the number of trials goes exponentially with the threshold t, one can see that if t is small it is easy for an adversary to get the correct passwords.

More importantly, the above attack implies that a legitimate client in C can also obtain all passwords of the other clients with the linear trials. Suppose that there are two legitimate clients C_1 and C_3 who make up a subgroup $SG = \{C_1, C_2, C_3\}$. After running the TA-PAKE protocol, as an adversary would do in the above, with server S, C_1 and C_3 can know the password of C_2 by checking possible N password candidates in the same way as above. By repeating this off-line dictionary attack $n-2$ times, C_1 and C_3 find out all passwords of the remaining clients in C.

3 The Model and Security Notions

In this section, we introduce the model based on [5,3], security notions and the underlying mathematical assumption.

The Model. We consider SG (i.e., a subgroup of C) and S as two parties that participate in the key exchange protocol P. Each of SG and S may have several instances called oracles involved in distinct, possibly concurrent, executions of P. We denote SG (resp., S) instances by SG^μ (resp., S^ν) where $\mu, \nu \in \mathbb{N}$, or by U in case of any instance. In the TAP protocol, each client C_i $(1 \leq i \leq n)$ of C and server S share a low-entropy secret pw_{C_i} drawn from a small dictionary of password $\mathbb{D}_{\mathsf{Password}}$, whose cardinality is N. Here we assume that an adversary $\mathcal{A}$ is not any client and server (i.e., $\mathcal{A} \notin \{C, S\}$). However, the adversary has the entire control of the network during the protocol execution which can be represented by allowing $\mathcal{A}$ to ask several queries to oracles. Let us show the capability of adversary $\mathcal{A}$ each query captures:

- $\mathsf{Execute}(SG^\mu, S^\nu)$: This query models passive attacks, where the adversary gets access to honest executions of P between the instances SG^μ and S^ν by eavesdropping.
- $\mathsf{Send}(U, m)$: This query models active attacks by having $\mathcal{A}$ send a message to instance U. The adversary $\mathcal{A}$ gets back the response U generates in processing the message m according to the protocol P. A query $\mathsf{Send}(SG^\mu, \mathtt{Start})$ initializes the key exchange protocol, and thus the adversary receives the first flow.
- $\mathsf{Reveal}(U)$: This query handles the misuse of the session key (e.g., use in a weak symmetric-key encryption) by any instance U. The query is only available to $\mathcal{A}$, if the instance actually holds a session key, and at that case the key is released to $\mathcal{A}$.
- $\mathsf{Test}(U)$: This oracle is used to see whether or not the adversary can obtain some information on the session key by giving a hint on the key. The Test-query can be asked at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the instance U is "fresh" in that the session key is not obviously known to the adversary. This query is answered as follows: one flips a private coin

$b \in \{0, 1\}$ and forwards the corresponding session key SK ($\mathsf{Reveal}(U)$ would output) if $b = 1$, or a random value with the same size except the session key if $b = 0$.

Security Notions. The adversary $\mathcal{A}$ is provided with random coin tosses, some oracles and then is allowed to invoke any number of queries as described above, in any order. The aim of the adversary is to break the privacy of the session key (a.k.a., semantic security) or the authentication of the parties in the context of executing P.

The AKE security is defined by the game $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$, in which the ultimate goal of the adversary is to guess the bit b involved in the Test-query by outputting this guess b'. We denote the AKE advantage, by $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$, as the probability that $\mathcal{A}$ can correctly guess the value of b. The protocol P is said to be (t, ε)-AKE-secure if $\mathcal{A}$'s advantage is smaller than ε for any adversary $\mathcal{A}$ running time t.

Another goal is to consider unilateral authentication of either SG ($\mathsf{SG\text{-}auth}$) or S ($\mathsf{S\text{-}auth}$) wherein the adversary impersonates a party. We denote by $\mathsf{Succ}_P^{\mathsf{SG-auth}}(\mathcal{A})$ (resp., $\mathsf{Succ}_P^{\mathsf{S-auth}}(\mathcal{A})$) the probability that $\mathcal{A}$ successfully impersonates an SG instance (resp., an S instance) in an execution of P, which means that S (resp., SG) agrees on a key while the latter is shared with no instance of SG (resp., S). A protocol P is said to be (t, ε)-Auth-secure if $\mathcal{A}$'s success probability for breaking either $\mathsf{SG\text{-}auth}$ or S-auth is smaller than ε for any adversary $\mathcal{A}$ running time t.

By following the definition of anonymity from [20], we can say that a protocol P is *anonymous* if a passive server cannot get any information about clients' identities (in SG) involved with the protocol, whereas the subgroup SG establishes a session key with the server. In other words, any subgroup can prove that it consists of legitimate members of the set C by sending its authenticator at the end of the protocol. Nevertheless, the server does not know who they are.

3.1 Computational Diffie-Hellman Assumption

A (t_1, ε_1)-$\mathsf{CDH}_{g,\mathbb{G}_p}$ attacker, in a finite cyclic group $\mathbb{G}_p$ of prime order p with g as a generator, is a probabilistic machine $\mathcal{B}$ running in time t_1 such that its success probability $\mathsf{Succ}_{g,\mathbb{G}_p}^{\mathsf{cdh}}(\mathcal{B})$, given random elements g^x and g^y to output g^{xy}, is greater than ε_1. We denote by $\mathsf{Succ}_{g,\mathbb{G}_p}^{\mathsf{cdh}}(t_1)$ the maximal success probability over every adversaries running within time t_1. The CDH-Assumption states that $\mathsf{Succ}_{g,\mathbb{G}_p}^{\mathsf{cdh}}(t_1) \leq \varepsilon_1$ for any t_1/ε_1 not too large.

4 A Secure Threshold Anonymous PAKE (TAP) Protocol

In this section, we propose a secure threshold anonymous PAKE (for short, TAP) protocol that has the following properties: 1) semantic security of session keys against an outside adversary; and 2) anonymity against a passive server, who follows the protocol honestly but is curious about clients' identities involved with the protocol. Here we assume that all clients C_i $(1 \leq i \leq n)$ of the set C has registered their passwords pw_{C_i} to a server S and the server holds the password verification data in an asymmetric form (i.e., $h^{\mathcal{F}(i,pw_{C_i})}$). For simplicity, we assign the clients consecutive integer i $(1 \leq i \leq n)$ where C_i can be regarded as the i-th client of C. In the TAP protocol, any subgroup SG consisting of at least t $(t > 1)$ clients wants to share a session key securely and anonymously with server S (see Fig. 2).

Rationale. A naive approach for secure threshold anonymous PAKE protocol is performing the existing (not threshold) anonymous PAKE protocol up to t times. This apparently entails a lot of messages to be exchanged between subgroup SG and server S. In order to construct efficiently, the TAP protocol has the following rationale. The first is that, instead of client's password itself, the output of $\mathcal{F}(i, pw_{C_i})$ is used as an exponent in order to compute the verification data W_i as in [20]. In fact, this plays a very important role when $t = 1$ (see Section 7) in that an adversary is enforced to make an on-line dictionary attack on a specific client, not the others. The second is that server generates only one Diffie-Hellman public value and its exponent is used to compute all of the possible Diffie-Hellman key K_j. As we will show in the proof, this is the reason why an adversary can get a factor n in the second term of the security result of Theorem 1. The third is that server sends $\{Z_j\}_{1 \leq j \leq n}$ by encrypting a share of the secret S with the hash of each Diffie-Hellman key. This is enough to guarantee clients' anonymity against an honest-but-curious server (see Theorem 2 in Section 5).

Step 1

1.1 Each client C_i, who belongs to the subgroup SG, chooses a random number x_i from $\mathbb{Z}_p^\star$ and computes the Diffie-Hellman public value $X_i \equiv g^{x_i}$. The client C_i also computes the password verification data $W_i \equiv h^{\mathcal{F}(i,pw_{C_i})}$ where i and pw_{C_i} are the index and the password, respectively, for C_i. The W_i is used to mask X_i, so that its resultant value X_i^* can be obtained in a way of $X_i^* \equiv X_i \times W_i$. The chosen x_i is kept secret by C_i.

1.2 By collaborating with one another, subgroup SG (or any client C_i) chooses $X_j^* \overset{R}{\leftarrow} \mathbb{G}_p$ for each C_j $(1 \leq j \neq i \leq n)$, who belongs to C but not to SG. Then the subgroup sends the threshold t, the masked public values $\{X_i^*\}_{1 \leq i \leq n}$, to the server, together with the set C of clients' identities.

Step 2

2.1 The server S chooses a random number y from $\mathbb{Z}_p^\star$ and a random secret S from $\mathbb{G}_p$, and computes its Diffie-Hellman public value $Y \equiv g^y$. The secret S is distributed as shares by using Shamir's (t, n) secret sharing scheme [15]. Specifically, server S generates the respective share $f(j)$, for all clients, from a polynomial $f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k$ with $u_0 = \mathsf{S}$ and coefficients u_k $(1 \leq k \leq t-1)$ randomly chosen from $\mathbb{G}_p$.

2.2 For the received X_j^* $(1 \leq j \leq n)$, server S computes $X_j \equiv X_j^*/W_j$ and the Diffie-Hellman key $K_j \equiv X_j^y$. The Z_j is derived from XORing S_j

Public information: $(\mathbb{G}_p, p, g, h), \mathcal{F}, \mathcal{G}, \mathcal{H}_1, \mathcal{H}_2, C = \{C_1, \cdots, C_n\}$

Subgroup $SG = \{\underbrace{C_1, C_4, \cdots, C_{n-3}}_{t}\}$

For each $C_i \in SG$,

$x_i \xleftarrow{R} \mathbb{Z}_p^\star$, $X_i \equiv g^{x_i}$,

$W_i \equiv h^{\mathcal{F}(i, pw_{C_i})}$,

and $X_i^* \equiv X_i \times W_i$.

For each $C_j \in C \backslash SG$ $(1 \le j \ne i \le n)$,

$X_j^* \xleftarrow{R} \mathbb{G}_p$.

$$\xrightarrow{C, t, \{X_i^*\}_{1 \le i \le n}}$$

Server S $\left(W_j \equiv h^{\mathcal{F}(j, pw_{C_j})},\ 1 \le j \le n\right)$

$y \xleftarrow{R} \mathbb{Z}_p^\star, Y \equiv g^y, \mathsf{S} \xleftarrow{R} \mathbb{G}_p$

$f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k$ where

$u_0 = \mathsf{S}$ and $u_k \xleftarrow{R} \mathbb{G}_p$

For $j = 1$ to n,

$\mathsf{S}_j \leftarrow f(j)$,

$X_j \equiv X_j^* / W_j$,

$K_j \equiv X_j^y$,

and $Z_j \leftarrow \mathcal{G}(j, K_j) \bigoplus \mathsf{S}_j$.

$V_S \leftarrow \mathcal{H}_1(C||S||Trans||\mathsf{S})$

$SK \leftarrow \mathcal{H}_2(C||S||Trans||\mathsf{S})$

$$\xleftarrow{S, Y, \{Z_j\}_{1 \le j \le n}, V_S}$$

For each $C_i \in SG$,

look for $Z_{j=i}$,

$K_i \equiv Y^{x_i}$ and $\mathsf{S}_i = Z_i \bigoplus \mathcal{G}(i, K_i)$.

$\mathsf{S}' = \sum_{k=1}^{t} \lambda_k \cdot \mathsf{S}_k$ where $\lambda_k \equiv \prod_{1 \le m \le t, m \ne k} \frac{m}{m-k}$

If $V_S \ne \mathcal{H}_1(C||S||Trans||\mathsf{S}')$, reject.

Otherwise, $SK \leftarrow \mathcal{H}_2(C||S||Trans||\mathsf{S}')$

and accept.

Fig. 2. A secure threshold anonymous PAKE (TAP) protocol where the threshold $t > 1$ and $Trans = t||\{X_i^*\}_{1 \le i \le n}||Y||\{Z_j\}_{1 \le j \le n}$

and the hashed output of index j and K_j: $Z_j \leftarrow \mathcal{G}(j, K_j) \bigoplus \mathsf{S}_j$ where $\mathsf{S}_j \leftarrow f(j)$.

2.3 Also server S generates an authenticator $V_S \leftarrow \mathcal{H}_1(C||S||t||\{X_i^*\}_{1 \le i \le n}|| Y||\{Z_j\}_{1 \le j \le n}||\mathsf{S})$ and a session key $SK \leftarrow \mathcal{H}_2(C||S||t||\{X_i^*\}_{1 \le i \le n}||Y|| \{Z_j\}_{1 \le j \le n}||\mathsf{S})$. Then the server sends its identity S, the Diffie-Hellman public value Y, $\{Z_j\}_{1 \le j \le n}$ and the authenticator V_S to subgroup SG.

Step 3

3.1 Each client C_i, who belongs to SG, first looks for $Z_{j=i}$ and computes the Diffie-Hellman key K_i with x_i: $K_i \equiv Y^{x_i}$. Now, client C_i extracts S_i from Z_i in an obvious way: $\mathsf{S}_i = Z_i \bigoplus \mathcal{G}(i, K_i)$.

3.2 By collaborating with one another, subgroup SG reconstructs S' from the t shares S_i by Lagrange interpolation: $\mathsf{S}' = \sum_{k=1}^{t} \lambda_k \cdot \mathsf{S}_k$ where $\lambda_k \equiv \prod_{1 \le m \le t, m \ne k} \frac{m}{m-k}$. If the received V_S is not valid (i.e., $V_S \ne \mathcal{H}_1(C||S||t|| \{X_i^*\}_{1 \le i \le n}||Y||\{Z_j\}_{1 \le j \le n}||\mathsf{S}'))$, the subgroup terminates the protocol. Otherwise, subgroup SG generates its session key $SK \leftarrow \mathcal{H}_2(C|| S||t|| \{X_i^*\}_{1 \le i \le n}||Y||\{Z_j\}_{1 \le j \le n}||\mathsf{S}')$. Obviously, any subgroup of less than t clients cannot generate a common session key SK.

Instead of collaborating with one another at Step 1.2 and 3.2, one client in the subgroup SG can choose X_j^* and reconstruct S' by collecting t shares from the other $t-1$ clients.

Remark 1. *In order to provide mutual authentication in the above protocol, we can simply add the subgroup's authenticator* $V_{SG} \leftarrow \mathcal{H}_3(C||S||t||\{X_i^*\}_{1\le i\le n}||Y|| \{Z_j\}_{1\le j\le n}||\mathsf{S})$*, as the third flow from subgroup SG to server S, before completing the* TAP *protocol. This is due to the well-known fact that the basic approach in the literature for adding authentication to an AKE protocol is to use the shared Diffie-Hellman key to construct a simple "authenticator" for the other party [5,3].*

5 Security

At first, we show that the TAP protocol of Fig. 2 distributes session keys that are semantically-secure and provides unilateral authentication of server S in the random oracle model [6].

Theorem 1. (AKE/UA Security) *Let P be the* TAP *protocol of Fig. 2 where passwords are independently chosen from a dictionary of size N and n is the number of clients such that $n \le 2\sqrt{N-1}-1$.*[3] *For any adversary $\mathcal{A}$ within a polynomial time t_1, with less than q_s active interactions with the parties (*Send*-queries), q_e passive eavesdroppings (*Execute*-queries) and asking q_f (resp., q_g) hash queries to $\mathcal{F}$ (resp., $\mathcal{G}$),* $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \le 4\varepsilon$ *and* $\mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) \le \varepsilon$*, with ε upper-bounded by*

$$\frac{3q_s}{N} + \frac{3nq_g^2}{2} \times \mathsf{Succ}_{g,\mathbb{G}_p}^{\mathsf{cdh}}(t_1 + 3\tau_e) + \frac{q_s}{2^{l_1}} + \frac{(q_e+q_s)^2}{|\mathbb{G}_p|^2} + \frac{q_f^2}{2p} + \frac{q_g^2 + 2(q_s+q_e)}{2|\mathbb{G}_p|} ,$$

where l_1 is the output length of $\mathcal{H}_1$ and τ_e denotes the computational time for an exponentiation in $\mathbb{G}_p$.

This theorem shows that the TAP protocol is secure against off-line dictionary attacks since the advantage of the adversary essentially grows with the ratio of interactions (number of Send-queries) to the number of passwords when $n \le 2\sqrt{N-1}-1$. We can easily see that the adversary gets a factor n in the second term since the server generates only one Diffie-Hellman public value and its exponent is used to compute all of the Diffie-Hellman keys K_j.[4] Due to the lack of space, we leave the proof in the full version of this paper.

Next we prove that the TAP protocol provides client's anonymity against a passive server.

Theorem 2. *The* TAP *protocol provides client's anonymity against a passive server in an information-theoretic sense.*

[3] In practice, $N = 2^{37}$ for MS-Windows passwords. It is sufficiently large for n.

[4] If we allow an adversary to corrupt up to $t-1$ clients, the factor n in the security result becomes 1.

Proof. Consider server S who follows the protocol honestly, but it is curious about clients' identities (in SG) involved with the TAP protocol. It is obvious that server S cannot get any information about SG's identities since, for each i $(1 \leq i \leq n)$, the X_i^* has a unique discrete logarithm of g and, with the randomly chosen x_i, it is the uniform distribution over $\mathbb{G}_p$. This also implies that the server cannot distinguish X_i^* (of $C_i \in SG$) from X_j^* (of $C_j \in C \backslash SG$) since they are completely independent one another. In addition, even if server S receives the subgroup's authenticator $V_{SG} \leftarrow \mathcal{H}_3(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S})$ at the end of the TAP protocol (in the case of mutual authentication), the $\{X_i^*\}_{1 \leq i \leq n}$ does not reveal any information about SG's identities from the fact that the probability for any subgroup, consisting of t or more than t clients, to compute S is equal. □

6 The Condition on *n*

Here we deduce the condition on n, appeared in Theorem 1, which is crucial in order to make the security result more meaningful. First, we give an informal definition of security against on-line dictionary attacks: a protocol is said to be secure against on-line dictionary attacks if an adversary can do no better than guess a password during each Send-query (i.e., an impersonation attack). However, the success probability of on-line attacks in the TAP protocol is greater than that in the 2-party PAKE protocols (see below).

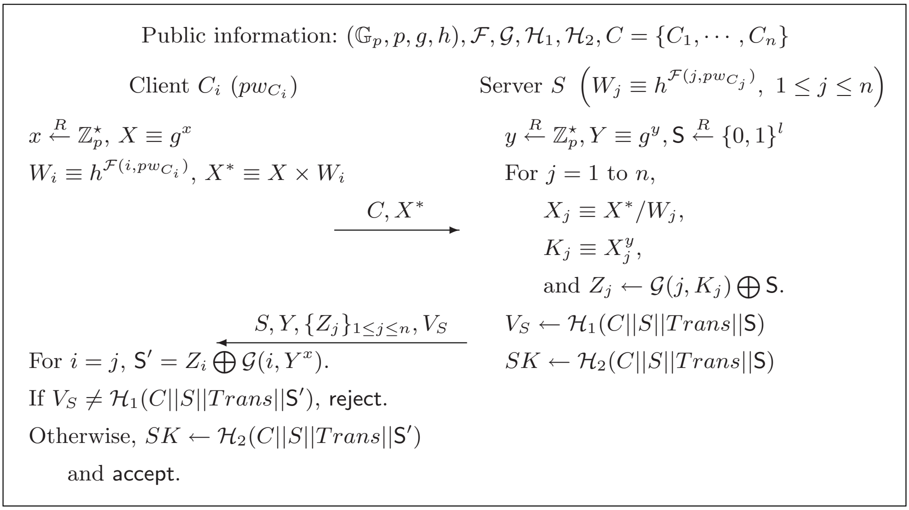
Theorem 3. *Consider an adversary who impersonates one party (i.e., subgroup SG or server S) for on-line dictionary attacks in the* TAP *protocol. Then the probability of the adversary is upper-bounded by*

$$\left\lceil \frac{n}{2} \right\rceil^2 \frac{1}{N(N-1)} \ .$$

Proof. When an adversary invokes Send-queries at **Game** $\mathbf{G}_5$ in the proof, we explain why the probability of on-line dictionary attacks is upper-bounded by the above. In order to maximize $\Pr[\mathsf{AskH1\text{-}WithSG}_5]$, the strategy the adversary can take is to first determine the threshold t and guess t passwords, each of which should be a password of one of n/t clients. Then the adversary sends the t and $\{X_i^*\}_{1 \leq i \leq n}$, as an honest party SG would do, to server S. After receiving the message from the server, the adversary can check whether the guessed passwords are correct or not by seeing the authenticator V_S. The maximal probability can be obtained when $t = 2$. That one password is correct with respect to $n/2$ clients happens with probability of $n/2N$. On the other hand, the probability for the other password is $n/2(N-1)$. For any n, one can get the upper-bound as above since the probability becomes smaller as t grows. As for $\Pr[\mathsf{AskH1\text{-}WithS}_5]$, the same discussion can be applied. □

Now the condition on n can be easily obtained by restricting the probability of Theorem 3 to $1/N$:

$$\left\lceil \frac{n}{2} \right\rceil^2 \frac{1}{N(N-1)} \leq \frac{(n+1)^2}{4N(N-1)} \leq \frac{1}{N} \ .$$



Public information: $(\mathbb{G}_p, p, g, h), \mathcal{F}, \mathcal{G}, \mathcal{H}_1, \mathcal{H}_2, C = \{C_1, \cdots, C_n\}$

Client C_i (pw_{C_i})

Server S $\left(W_j \equiv h^{\mathcal{F}(j, pw_{C_j})},\ 1 \le j \le n\right)$

$x \xleftarrow{R} \mathbb{Z}_p^\star,\ X \equiv g^x$

$W_i \equiv h^{\mathcal{F}(i, pw_{C_i})},\ X^* \equiv X \times W_i$

$\xrightarrow{C, X^*}$

$y \xleftarrow{R} \mathbb{Z}_p^\star, Y \equiv g^y, \mathsf{S} \xleftarrow{R} \{0,1\}^l$

For $j = 1$ to n,

$X_j \equiv X^*/W_j$,

$K_j \equiv X_j^y$,

and $Z_j \leftarrow \mathcal{G}(j, K_j) \bigoplus \mathsf{S}$.

$\xleftarrow{S, Y, \{Z_j\}_{1 \le j \le n}, V_S}$

$V_S \leftarrow \mathcal{H}_1(C||S||Trans||\mathsf{S})$

$SK \leftarrow \mathcal{H}_2(C||S||Trans||\mathsf{S})$

For $i = j$, $\mathsf{S}' = Z_i \bigoplus \mathcal{G}(i, Y^x)$.

If $V_S \neq \mathcal{H}_1(C||S||Trans||\mathsf{S}')$, reject.

Otherwise, $SK \leftarrow \mathcal{H}_2(C||S||Trans||\mathsf{S}')$

and accept.

Fig. 3. An efficient anonymous PAKE protocol when $t = 1$ and where $Trans = X^*||Y||\{Z_j\}_{1 \le j \le n}$

7 When the Threshold $t = 1$

If we only consider a passive server in an anonymous PAKE protocol, an efficient construction for the threshold $t = 1$ can be easily derived from the TAP protocol (see Fig. 3). The main modification from the TAP protocol is that client C_i only computes his masked Diffie-Hellman public value X^* and the hash function $\mathcal{G}$ has the range of $\{0,1\}^l$. In fact, the resultant protocol can be seen as another OT protocol for the case that all the messages of sender is the same. By following the security proof, we can remove the condition on n because the on-line attacks at **Game** $\mathbf{G}_5$ is limited to one client.

Here, we show how much our protocol of Fig. 3 is efficient compared to the original (not threshold) anonymous PAKE protocol (in Section 3.2 of [20]) in terms of computation costs and communication bandwidth to be required (see Table 1 and 2). In general, the number of modular exponentiations is a major factor to evaluate efficiency of a cryptographic protocol because that is the most power-consuming operation. So we count the number of modular exponentiations as computation costs of client C_i and server S. The figures in the parentheses are the remaining number of modular exponentiations after excluding those that are pre-computable. In terms of communication bandwidth, $|\cdot|$ indicates its bit-length and hash denotes hash functions.

With respect to computation costs in our protocol, client C_i (resp., server S) is required to compute 3 (resp., $n + 1$) modular exponentiations. When pre-computation is allowed, the remaining costs of client C_i (resp., server S) are 2 (resp., n) modular exponentiations. One can easily see that our protocol has more than 50% reduction from the APAKE protocol in the number of modu-

Table 1. Comparison of anonymous PAKE protocols as for computation costs where n is the number of clients

	The number of modular exponentiations	
Protocols	Client C_i	Server S
APAKE [20]	6 (4)	$4n+2$ $(3n+1)$
Our protocol of Fig. 3	3 (2)	$n+1$ (n)

Table 2. Comparison of anonymous PAKE protocols as for communication bandwidth where n is the number of clients

Protocols	Communication bandwidth
APAKE [20]	$\lvert C\rvert + \lvert S\rvert + (n+1)\lvert \mathsf{hash}\rvert + (n+2)\lvert p\rvert$
Our protocol of Fig. 3	$\lvert C\rvert + \lvert S\rvert + (n+1)\lvert \mathsf{hash}\rvert + 2\lvert p\rvert$

lar exponentiations for both client and server. With respect to communication bandwidth, our protocol requires a bandwidth of $((n+1)|\mathsf{hash}| + 2|p|)$-bits except the length of identities C and S where the bandwidth for the modulus size $|p|$ is independent from the number of clients while the APAKE protocol is not. Let us consider the minimum security parameters recommended in practice ($|p| = 1024$ and $|\mathsf{hash}| = 160$). The gap of communication bandwidths between our and APAKE protocols becomes larger as the number of clients increases.

8 Conclusions

After showing insecurity of the previous threshold anonymous PAKE protocol, we have proposed a secure construction (the TAP protocol) which provides not only semantic security of session keys but also anonymity against a passive server. We also proved its security of the TAP protocol in the random oracle model with the reduction to the computational Diffie-Hellman problem. Moreover, we showed the condition on n in order to get the optimal security result against on-line dictionary attacks. For the threshold $t = 1$, we have proposed an efficient anonymous PAKE protocol that can be obtained by slightly modifying the TAP protocol. The resultant protocol significantly improves efficiency in terms of computation costs and communication bandwidth compared to the original (not threshold) anonymous PAKE protocol [20].

References

1. Abdalla, M., Pointcheval, D.: Simple Password-Based Encrypted Key Exchange Protocols. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
2. Bellare, M., Canetti, R., Krawczyk, H.: A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In: Proc. of 30th ACM Symposium on Theory of Computing (STOC), pp. 419–428. ACM Press, New York (1998)

3. Bresson, E., Chevassut, O., Pointcheval, D.: New Security Results on Encrypted Key Exchange. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 145–158. Springer, Heidelberg (2004)
4. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In: Proc. of IEEE Symposium on Security and Privacy, pp. 72–84 (1992)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Proc. of ACM CCS 1993, pp. 62–73 (1993)
7. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
8. Chu, C.K., Tzeng, W.G.: Efficient k-Out-of-n Oblivious Transfer Schemes with Adaptive and Non-adaptive Queries. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 172–183. Springer, Heidelberg (2005)
9. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
10. Diffie, W., van Oorschot, P., Wiener, M.: Authentication and Authenticated Key Exchange. In: Proc. of Designs, Codes, and Cryptography, pp. 107–125 (1992)
11. http://grouper.ieee.org/groups/1363/passwdPK/submissions.html
12. Krawczyk, H.: SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003)
13. MacKenzie, P., Shrimpton, T., Jakobsson, M.: Threshold Password-Authenticated Key Exchange. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 385–400. Springer, Heidelberg (2002)
14. Nguyen, M.H.: The Relationship Between Password-Authenticated Key Exchange and Other Cryptographic Primitives. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 457–475. Springer, Heidelberg (2005)
15. Shamir, A.: How to Share a Secret. Proc. of Communications of the ACM 22(11), 612–613 (1979)
16. Shoup, V.: On Formal Models for Secure Key Exchange. IBM Research Report RZ 3121, (1999) Available at http://eprint.iacr.org/1999/012
17. Shoup, V.: OAEP Reconsidered. Journal of Cryptology 15(4), 223–249 (2002)
18. Shoup, V.: Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive: Report 2004 /332, Available at http://eprint.iacr.org/2004/332
19. Tzeng, W.G.: Efficient 1-Out-n Oblivious Transfer Schemes. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 159–171. Springer, Heidelberg (2002)
20. Viet, D.Q., Yamamura, A., Tanaka, H.: Anonymous Password-Based Authenticated Key Exchange. In: Maitra, S., Madhavan, C.E.V., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 244–257. Springer, Heidelberg (2005)
21. Viet, D.Q., Yamamura, A., Tanaka, H.: Anonymous Password-Based Authenticated Key Exchange. In: Proc. of the 2006 Symposium on Cryptography and Information Security (SCIS 2006), 3D3-4 (January 2006)
22. Wilson, S.B., Johnson, D., Menezes, A.: Key Agreement Protocols and their Security Analysis. In: Proc. of IMA International Conference on Cryptography and Coding (December 1997)

Author Index