Lecture Notes in Computer Science 2846

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Jianying Zhou Moti Yung Yongfei Han (Eds.)

Applied Cryptography and Network Security

First International Conference, ACNS 2003
Kunming, China, October 16-19, 2003
Proceedings

Springer

Volume Editors

Jianying Zhou
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
E-mail: jyzhou@i2r.a-star.edu.sg

Moti Yung
Columbia University
S.W. Mudd Building, Computer Science Department
New York, NY 10027, USA
E-mail: moti@cs.columbia.edu

Yongfei Han
ONETS, Shangdi Zhongguancun Chuangye Dasha
Haidian District, Beijing 100085, China
E-mail: yongfei_han@onets.com.cn

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): E.3, C.2, D.4.6, H.3-4, K.4.4, K.6.5

ISSN 0302-9743
ISBN 3-540-20208-0 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin GmbH
Printed on acid-free paper SPIN 10960585 06/3142 5 4 3 2 1 0

Preface

The 1st International Conference on "Applied Cryptography and Network Security" (ACNS 2003) was sponsored and organized by ICISA (International Communications and Information Security Association), in cooperation with MiAn Pte. Ltd. and the Kunming government. It was held in Kunming, China in October 2003. The conference proceedings was published as Volume 2846 of the Lecture Notes in Computer Science (LNCS) series of Springer-Verlag.

The conference received 191 submissions, from 24 countries and regions; 32 of these papers were accepted, representing 15 countries and regions (acceptance rate of 16.75%). In this volume you will find the revised versions of the accepted papers that were presented at the conference. In addition to the main track of presentations of accepted papers, an additional track was held in the conference where presentations of an industrial and technical nature were given. These presentations were also carefully selected from a large set of presentation proposals.

This new international conference series is the result of the vision of Dr. Yongfei Han. The conference concentrates on current developments that advance the areas of applied cryptography and its application to systems and network security. The goal is to represent both academic research works and developments in industrial and technical frontiers. We thank Dr. Han for initiating this conference and for serving as its General Chair.

Many people and organizations helped in making the conference a reality. We thank the conference sponsors: the Kunming government, MiAn Pte. Ltd., and ICISA. We greatly thank the organizing committee members for taking care of the registration, logistics, and local arrangements. It is due to their hard work that the conference was possible. We also wish to thank Springer and Mr. Alfred Hofmann and his staff for the advice regarding the publication of the proceedings as a volume of LNCS. Our deepest thanks go to the program committee members for their hard work in reviewing papers. We also wish to thank the external reviewers who assisted the program committee members.

Last, but not least, special thanks are due to all the authors who submitted papers and to the conference participants from all over the world. We are very grateful for their support, which was especially important in these difficult times when the SARS outbreak impacted many countries, especially China. It is in such challenging times for humanity that the strength and resolve of our community is tested: the fact that we were able to attract many papers and prepare and organize this conference is testament to the determination and dedication of the cryptography and security research community worldwide.

October 2003

Jianying Zhou
Moti Yung

ACNS 2003

1st International Conference on Applied Cryptography and Network Security

Kunming, China
October 16–19, 2003

Sponsored and organized by

International Communications and Information Security Association (ICISA)

In co-operation with

MiAn Pte. Ltd. (ONETS), China
and
Kunming Government, China

General Chair

Yongfei Han ..... ONETS, China

Program Chairs

Jianying Zhou ..... Institute for Infocomm Research, Singapore
Moti Yung ..... Columbia University, USA

Program Committee

Thomas Berson ..... Anagram, USA
Robert Deng ..... Institute for Infocomm Research, Singapore
Xiaotie Deng ..... City University, Hong Kong
Dengguo Feng ..... Chinese Academy of Sciences, China
Shai Halevi ..... IBM T.J. Watson Research Center, USA
Amir Herzberg ..... Bar-Ilan University, Israel
Sushil Jajodia ..... George Mason University, USA
Markus Jakobsson ..... RSA Lab, USA
Kwangjo Kim ..... Information and Communications University, Korea
Kwok-Yan Lam ..... Tsinghua University, China
Javier Lopez ..... University of Malaga, Spain
Keith Martin ..... Royal Holloway, University of London, UK
Catherine Meadows ..... Naval Research Lab, USA
Chris Mitchell ..... Royal Holloway, University of London, UK

Atsuko Miyaji ..... JAIST, Japan
David Naccache ..... Gemplus, France
Kaisa Nyberg ..... Nokia, Finland
Eiji Okamoto ..... University of Tsukuba, Japan
Rolf Oppliger ..... eSECURITY Technologies, Switzerland
Susan Pancho ..... University of the Philippines, Philippines
Guenther Pernul ..... University of Regensburg, Germany
Josef Pieprzyk ..... Macquarie University, Australia
Bart Preneel ..... K.U. Leuven, Belgium
Sihan Qing ..... Chinese Academy of Sciences, China
Leonid Reyzin ..... Boston University, USA
Bimal Roy ..... Indian Statistical Institute, India
Kouichi Sakurai ..... Kyushu University, Japan
Pierangela Samarati ..... University of Milan, Italy
Gene Tsudik ..... University of California, Irvine, USA
Wen-Guey Tzeng ..... National Chiao Tung University, Taiwan
Vijay Varadharajan ..... Macquarie University, Australia
Adam Young ..... Cigital, USA
Yuliang Zheng ..... University of North Carolina, Charlotte, USA

Organizing Committee

Yongfei Han ..... ONETS, China
Chuankun Wu ..... Chinese Academy of Sciences, China
Li Xu ..... ONETS, China

External Reviewers

Aditya Bagchi, Antoon Bosselaers, Christain Breu, Christophe De Cannière, Xiaofeng Chen, Benoit Chevallier-Mames, Siu-Leung Chung, Tanmoy Kanti Das, Mike David, Xuhua Ding, Ratna Dutta, Matthias Fitzi, Jacques Fournier, Youichi Futa, Hossein Ghodosi, Pierre Girard, Zhi Guo, Michael Hitchens, Kenji Imamoto, Sarath Indrakanti, Gene Itkis, Hiroaki Kikuchi, Svein Knapskog, Bao Li, Tieyan Li, Dongdai Lin, Wenqing Liu, Anna Lysyanskaya, Hengtai Ma, Subhamoy Maitra, Kostas Markantonakis, Eddy Masovic, Mitsuru Matusi, Pradeep Mishra, Sourav Mukherjee, Bjoern Muschall, Einar Mykletun, Mridul Nandy, Maithili Narasimha, Svetla Nikova, Pascal Paillier, Pinakpani Pal, Kenny Paterson, Stephanie Porte, Geraint Price, Torsten Priebe, Michael Quisquater, Pankaj Rohatgi, Ludovic Rousseau, Craig Saunders, Jasper Scholten, Yaron Sella, Hideo Shimizu, Igor Shparlinski, Masakazu Soshi, Ron Steinfeld, Hongwei Sun, Michael Szydlo, Uday Tupakula, Guilin Wang, Huaxiong Wang, Mingsheng Wang, Christopher Wolf, Hongjun Wu, Wenling Wu, Yongdong Wu, Shouhuai Xu, Masato Yamamichi, Jeong Yi, Xibin Zhao

Table of Contents

Security Modelling

Web Security

Security Protocols

Cryptanalysis

Key Management

Efficient Implementations

Multi-party Computation from Any Linear Secret Sharing Scheme Unconditionally Secure against Adaptive Adversary: The Zero-Error Case

Ventzislav Nikov[1], Svetla Nikova[2⋆], and Bart Preneel[2]

[1] Department of Mathematics and Computing Science,
Eindhoven University of Technology
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
v.nikov@tue.nl
[2] Department Electrical Engineering, ESAT/COSIC,
Katholieke Universiteit Leuven, Kasteelpark Arenberg 10,
B-3001 Heverlee-Leuven, Belgium
{svetla.nikova,bart.preneel}@esat.kuleuven.ac.be

Abstract. We consider a generalized adaptive and active adversary model for unconditionally secure Multi-Party Computation (MPC) in the zero error case.

Cramer *et al.* proposed a generic approach to build a *multiplicative* Monotone Span Programs (MSP) – the special property of a Linear Secret Sharing Schemes (LSSS) that is needed to perform a multiplication of shared values. They give an efficient generic construction to build verifiability into every LSSS and to obtain from any LSSS a multiplicative LSSS for the same access structure. But the multiplicative property guarantees security against passive adversary only. For an active adversary a strong multiplicative property is required. Unfortunately there is no known efficient construction to obtain a strongly multiplicative LSSS yet.

Recently Nikov *et al.* have expanded the construction of Cramer *et al.* using a different approach. Multiplying two different MSP M_1 and M_2 computing the access structures Γ_1 and Γ_2 a new MSP M called "resulting" is obtained. M computes a new access structure $\Gamma \subset \Gamma_1 \quad (or \Gamma_2)$. The goal of this construction is to enable the investigation of how the properties that Γ should fulfil are linked to the initial access structures Γ_1 and Γ_2. It is proved that Γ_2 should be a dual access structure of Γ_1 in order to have a multiplicative resulting MSP. But there are still not known requirements for initial access structures in order to obtain strongly multiplicative resulting MSP. Nikov *et al.* proved that to have unconditionally secure MPC the following minimal conditions for the resulting access structure should be satisfied $(\Gamma_A \uplus \Gamma_A)^{\perp} \subseteq \Gamma$.

In this paper we assume that the resulting MSP could be constructed such that the corresponding access structure Γ satisfies the required

⋆ The author was partially supported by IWT and Concerted Research Action GOA-MEFISTO-666 of the Flemish Government

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 1–15, 2003.

properties. Our goal is to study the requirements that Γ should fulfil in order to have an MPC unconditionally secure against adaptive and active adversary in the zero error case. First, we prove that Γ could satisfy weaker conditions than those in Nikov *et al.*, namely $\Gamma_A^{\perp} \subseteq \Gamma$. Second, we propose a commitment "degree reduction" protocol which allows the players to "reduce" one access structure, e.g. Γ, to another access structure Γ_3. This reduction protocol appears to be a generalization of the reduction protocol of Cramer *et al.* in the sense that we can choose to reduce Γ to the initial access structures Γ_1 or Γ_2, or to a new one Γ_3. This protocol is also more efficient, since it requires less Verifiable Secret Sharing Schemes to be used.

Keywords: general secure multi-party computation, verifiable secret sharing, linear secret sharing, monotone span programs, general adversaries, information theoretic security.

1 Introduction

Secure *multi-party computation* (MPC) can be defined as follows: n players compute an agreed function of their inputs in a "secure" way, where "secure" means guaranteeing the correctness of the output as well as the privacy of the players' inputs, even when some players cheat. A key tool for secure MPC, is the *verifiable secret sharing* (VSS) [6,1]. In VSS a dealer distributes a secret value among the players, where the dealer and/or some of the players may be cheating. It is guaranteed that if the dealer is honest, then the cheaters obtain no information about the secret, and all honest players will later be able to reconstruct it, without the help of the dealer. Even if the dealer cheats, a unique value will be determined and is reconstructible without the cheaters' help.

In [18] Shamir introduced the concept of *secret sharing* as a tool to protect a secret simultaneously from exposure and from being lost. It allows a so called *dealer* to share the secret among a set of entities, usually called *players*, in such a way that only certain specified subsets of the players are able to reconstruct the secret while smaller subsets have no information about it. The groups who are allowed to reconstruct the secret are called *qualified*, and the groups who should not be able to obtain any information about the secret are called *forbidden*. The collection of all qualified groups is denoted by Γ, and the collection of all forbidden groups is denoted by Δ. The tuple (Γ, Δ) is called an *access structure* if $\Gamma \cap \Delta = \emptyset$. Denote by $P = \{P_1, \ldots, P_n\}$ the set of participants in the scheme and by $\mathcal{P}(P)$ the set of all subsets of P. If $\Gamma \cup \Delta = \mathcal{P}(P)$, i.e., $\Gamma = \Delta^c$ is the complement of Δ, then (Γ, Δ) is *complete* and it is denoted simply by Γ. When Γ is complete the SSS is called perfect.

Usually the cheating is represented as an *adversary* who may corrupt some subset of the players. One can distinguish between *passive* and *active* corruption, see Fehr and Maurer, [8] for recent results. Passive corruption means that the adversary obtains the complete information held by the corrupt players, but the players execute the protocol correctly. Active corruption means that the adversary takes full control of the corrupt players. Active corruption is strictly stronger

than passive corruption. The adversary is characterized by a *privacy structure* Δ and an *adversary structure* $\Delta_A \subseteq \Delta$. Denote the complement $\Gamma_A = \Delta_A^c$ and call its dual access structure $\Gamma_A^\perp$ the *honest* (or *good*) players structure. Both passive and active adversaries may be *static*, meaning that the set of corrupt players is chosen once and for all before the protocol starts, or *adaptive* meaning that the adversary can at any time during the protocol choose to corrupt a new player based on all the information he has at the time, as long as the total set is in Δ_A. Most proposed Secret Sharing Schemes (SSS) are *linear*, but the concept of a Linear Secret Sharing Scheme (LSSS) was first considered in its full generality by Karchmer and Wigderson in [13], who introduced the equivalent notion of *Monotone Span Program* (MSP), which we describe later. Each linear SSS can be viewed as derived from a monotone span program $\mathcal{M}$ computing its access structure. On the other hand, each monotone span program gives rise to an LSSS. Hence, one can identify an LSSS with its underlying monotone span program. Such an MSP always exists, because MSPs can compute any monotone function. Since an LSSS neither guarantees reconstructability when some shares are incorrect, nor verifiability of a shared value the stronger primitive – Verifiable Secret Sharing has been introduced.
We will consider any complete general monotone access structure Γ, which describes subsets of participants that are qualified to recover the secret $s \in \mathbb{F}$ ($\mathbb{F}$ here is a finite field) in the set of possible secret values, as long as it admits a linear secret sharing scheme. We will consider also the standard *synchronous model* with a *broadcast channel*.

1.1 Related Work

This subsection contains some basic definitions, notations and results. For an arbitrary matrix M over $\mathbb{F}$, with m rows labelled by $1, \ldots, m$ let M_A denote the matrix obtained by keeping only those rows i with $i \in A$, where A is an arbitrary non-empty subset of $\{1, \ldots, m\}$. If $\{i\} = A$ we write M_i. Let M_A^T denote the transpose of M_A, and let $Im(M_A^T)$ denote the $\mathbb{F}$-linear span of the rows of M_A. We use $Ker(M_A)$ to denote the kernel of M_A, i.e., all linear combinations of the columns of M_A, leading to 0.
Let $v = (v_1, \ldots, v_{t_1}) \in \mathbb{F}^{t_1}$ and $w = (w_1, \ldots, w_{t_2}) \in \mathbb{F}^{t_2}$ be two vectors. The tensor vector product $v \otimes w$ is defined as a vector in $\mathbb{F}^{t_1 t_2}$ such that the j-coordinate in v (denoted by v_j) is replaced by $v_j w$, i.e., $v \otimes w = (v_1 w, \ldots, v_{t_1} w) \in \mathbb{F}^{t_1 t_2}$. The Kronecker product of matrices is defined as tensor vector multiplication of each row from the first matrix to each row from the second matrix.

Definition 1. *[5] The dual $\Gamma^\perp$ of a monotone access structure Γ defined on P is the collection of sets $A \subseteq P$ such that $A^c \notin \Gamma$.*

The following operation (called element-wise union) for monotone decreasing (increasing) sets was introduced in [15,8].

Definition 2. *For monotone decreasing sets Δ_1, Δ_2 and for monotone increasing sets Γ_1, Γ_2, all defined for the same set of participants, the* element-wise union *operation $*$ is defined by:*

$$\Delta_1 * \Delta_2 = \{A_1 \cup A_2; A_1 \in \Delta_1, A_2 \in \Delta_2\},$$
$$\textit{resp.} \qquad \Gamma_1 * \Gamma_2 = \{A_1 \cup A_2; A_1 \notin \Gamma_1, A_2 \notin \Gamma_2\}^c.$$

Throughout the paper we will consider presence of adaptive adversary. Let Q^2, resp. Q^3 be the conditions on an adversary structure that *no two*, resp. *no three* of the sets in the structure cover the full players set P. The adversary that we tolerate is at least a Q^2 (resp. Q^3) adversary in the passive (resp. active) scenario (see [12,4]). Since the condition Q^2 is equivalent to $\Delta_A \cap \Gamma_A^{\perp} = \emptyset$ (i.e., $\Gamma_A^{\perp} \subseteq \Gamma_A$), the honest players structure has no intersection with the adversary structure.
Recently Maurer [14] proved that general perfect information-theoretically secure MPC secure against a (Δ_1, Δ_A)-adversary is possible if and only if $P \notin \Delta_1 \uplus \Delta_1 \uplus \Delta_A$ or equivalently, if and only if $\Gamma_A^{\perp} \subseteq \Gamma_1 \uplus \Gamma_1$. Maurer consider the case, when the secrets are shared using only one MSP. Notice that thanks to the local computation model for MPC the interaction between players is reduced, and in this way we may think of the MPC as a kind of VSS.
A recent result, which gives necessary and sufficient conditions for the existence of information-theoretically secure VSS has been presented by Fehr and Maurer in [8]. They prove that the robustness conditions for VSS are fulfilled if and only if $P \notin \Delta \uplus \Delta_A \uplus \Delta_A$ or equivalently, if and only if $(\Gamma_A \uplus \Gamma_A)^{\perp} \subseteq \Gamma$.
As mentioned earlier, MSPs are essentially equivalent to LSSS's (see e.g. [13]). It turns out to be convenient to describe our protocols in terms of MSPs as we will do for the rest of the paper. A formal definition for an MSP follows.

Definition 3. *[3,4] A* **Monotone Span Program** *(MSP) $\mathcal{M}$ is a quadruple $(\mathbb{F}, M, \varepsilon, \psi)$, where $\mathbb{F}$ is a finite field, M is a matrix (with m rows and $d \leq m$ columns) over $\mathbb{F}$, $\psi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ is a surjective function and ε is a fixed vector, called target vector, e.g. column vector $(1, 0, ..., 0) \in \mathbb{F}^d$. The size of $\mathcal{M}$ is the number m of rows.*

As ψ labels each row with a number from $[1, \ldots, m]$ corresponding to a fixed player, we can think of each player as being the "owner" of one or more rows. For every player we consider a function φ which gives the set of rows owned by the player, i.e., φ is (in some sense) inverse of ψ.
An MSP is said to compute a (complete) access structure Γ when $\varepsilon \in Im(M^T_{\varphi(G)})$ if and only if G is a member of Γ. Hence, the players can reconstruct the secret precisely if the rows they own contain in their linear span the target vector of $\mathcal{M}$, and otherwise they get no information about the secret, i.e., there exists a so called *recombination vector* $\mathbf{r}$ such that $\langle \mathbf{r}, M_{\varphi(G)}(s, \rho) \rangle = s$ and $M^T_{\varphi(G)}\mathbf{r} = \varepsilon$ for any secret s and any ρ. It is well known that the vector $\varepsilon \notin Im(M_N^T)$ if and only if there exists a $\mathbf{k} \in \mathbb{F}^d$ such that $M_N\mathbf{k} = 0$ and $\mathbf{k}_1 = 1$.
The main goal of our paper is to study the properties of a construction which builds MPCs from any LSSS. It is well known that because of the linearity the LSSS provides it is easy to add secrets securely. Therefore to achieve general

MPC, it suffices to implement multiplication of shared secrets. That is, we need a protocol where each player initially holds shared secrets s and s', and ends up holding a share of the product ss'. Several such protocols are known for the threshold case [1,2,10,11] and for general access structure [3,4,17].
We follow the approach proposed by Cramer *et al.* in [3,4] to build an MPC from any LSSS, provided that the LSSS is what is called *(strongly) multiplicative*. Loosely speaking, an LSSS is (strongly) multiplicative if each player P_i can compute from his shares (of secrets s and s') a value c_i, such that the product ss' can be obtained using all values (only values from honest players).
In a recent paper by Nikov *et al.* [17] the $\diamond$ construction for multiplying two MSPs has been proposed. Let Γ_1 and Γ_2 be access structures, computed by MSPs $\mathcal{M}_1 = (\mathbb{F}, M_1, \varepsilon_1, \psi_1)$ and $\mathcal{M}_2 = (\mathbb{F}, M_2, \varepsilon_2, \psi_2)$. Let also M_1 be an $m_1 \times d_1$ matrix, M_2 be an $m_2 \times d_2$ matrix and φ_1, φ_2 be the "inverse" functions of ψ_1 and ψ_2. Consider the vector x. The coordinates in x, which belong to the player t are collected in a sub-vector x_t or $x = (\bar{x}_1, \ldots, \bar{x}_n)$. First the operation $\diamond$ for vectors is defined as follows:

$$x \diamond y = (\bar{x}_1 \otimes \bar{y}_1, \ldots, \bar{x}_n \otimes \bar{y}_n).$$

Denote by $(M_1)_{\mathbf{t}}$ the matrix formed by rows of M_1 owned by the player t and correspondingly by $(M_2)_{\mathbf{t}}$ the matrix formed by rows of M_2 owned by the same player. Hence M_1 can be presented as a concatenation of the matrices $(M_1)_{\mathbf{t}}$ for $t = 1, \ldots, n$. Then the operation $\diamond$ for matrices is defined as the concatenation of matrices $(M_1)_{\mathbf{t}} \otimes (M_2)_{\mathbf{t}}$ for $t = 1, \ldots, n$, i.e.,

$$M = M_1 \diamond M_2 = \begin{pmatrix} (M_1)_{\mathbf{1}} \otimes (M_2)_{\mathbf{1}} \\ \ldots \\ (M_1)_{\mathbf{n}} \otimes (M_2)_{\mathbf{n}} \end{pmatrix}.$$

Finally, the operation $\diamond$ for two MSP could be defined as:

Definition 4. *[17] Define MSP $\mathcal{M}$ to be $(\mathbb{F}, M = M_1 \diamond M_2, \varepsilon = \varepsilon_1 \diamond \varepsilon_2, \psi)$, where $\psi(i,j) = r$ if and only if $\psi_1(i) = \psi_2(j) = r$ and the size of $\mathcal{M}$ is $m = \sum_i |\varphi_1(i)||\varphi_2(i)| = \sum_i |\varphi(i)|$. Given two MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$, the MSP $\mathcal{M}$ is called their* **multiplicative resulting MSP** *and denoted by $\mathcal{M} = \mathcal{M}_1 \diamond \mathcal{M}_2$ if there exists an m-vector $\mathbf{r}$ called a recombination vector, such that for any two secrets s' and s'' and any ρ' and ρ'', it holds that*

$$s's'' = \langle \mathbf{r}, M_1(s', \rho') \diamond M_2(s'', \rho'') \rangle = \langle \mathbf{r}, M((s', \rho') \otimes (s'', \rho'')) \rangle .$$

The MSP $\mathcal{M}$ is called their **strongly multiplicative resulting MSP** *if the access structure Γ computed by $\mathcal{M}$ is such that for any players' subset $A \in \Gamma$, $\mathcal{M}_A$ is the multiplicative resulting MSP of $(\mathcal{M}_1)_A$ and $(\mathcal{M}_2)_A$.*

The last definition means that one can construct a strongly multiplicative resulting MSP, computing the product of the secrets shared by MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$, with some access structure Γ. The difference between the multiplicative resulting MSP and the strongly multiplicative resulting MSP is that in the first case $\Gamma = \{P\}$.

It has been proved in [17] that $\Gamma \subseteq \Gamma_1 \uplus \Gamma_2$. In the model of MPC proposed in [17] the secrets are shared using VSS and two MSP $\mathcal{M}_1$ and $\mathcal{M}_2$. Hence the adaptive adversary has two privacy structures Δ_1, Δ_2 and one adversary structure $\Delta_A \subseteq \Delta_1$, $\Delta_A \subseteq \Delta_2$. Such an adversary is denoted by $(\Delta_1, \Delta_2, \Delta_A)$-adversary.
In the computational model for MPC the authors in [17] propose the so called "algebraic simplification for multiplication" protocol which uses homomorphic commitments in the strongly multiplicative case of general MPC. In fact, the "algebraic simplification for multiplication" protocol allows the players to "reduce" one access structure Γ to another access structure Γ_3, provided that the VSS conditions for Γ_3 hold. As it is proved in [17] to build a MPC protocol secure against an adaptive adversary in the computational model it is sufficient the MSPs $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ to satisfy the VSS conditions, i.e., $\Gamma_A^{\perp} \subseteq \Gamma_i$ for $i = 1, 2, 3$; $\mathcal{M}$ to be resulting MSP of $\mathcal{M}_1$ and $\mathcal{M}_2$, i.e., $\Gamma \subseteq \Gamma_1 \uplus \Gamma_2$ and Γ to satisfy the strong multiplicative property, i.e., $\Gamma_A^{\perp} \subseteq \Gamma$. On the other hand the lack of "algebraic simplification for multiplication" protocol in the information-theoretic scenario impose stronger conditions for the strongly multiplicative case of general MPC. It is proved in [17] that it is sufficient for the MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$ to satisfy the VSS conditions from [8], i.e., $(\Gamma_A \uplus \Gamma_A)^{\perp} \subseteq \Gamma_i$ for $i = 1, 2$; $\mathcal{M}$ to be resulting MSP of $\mathcal{M}_1$ and $\mathcal{M}_2$, i.e., $\Gamma \subseteq \Gamma_1 \uplus \Gamma_2$ and Γ to satisfy the following property,

$$(\Gamma_A \uplus \Gamma_A)^{\perp} \subseteq \Gamma. \tag{1}$$

1.2 Results of This Paper

The condition (1) is sufficient to multiply securely two secrets, but it is insufficient to perform general MPC, since with each multiplication the access structure Γ becomes "smaller" and "smaller". Hence besides multiplying securely we need a "degree reduction" protocol to "reduce" the access structure Γ to another access structure e.g. Γ_3. The solution that we propose is parallel to the one in the threshold case, where after multiplication we have threshold $2t$ and reduce it to threshold t as Ben-Or *et al.* show in [1].
In this paper we build an information-theoretically secure simplification protocol for multiplication, which is an important step in order to be achieved general secure MPC. The main hurdle to overcome in the "degree reduction" protocol is the additional check which ensures the commitment to the re-shared shares. The clue in this additional check is the change of the basis (see Section 3.3).
Our main result follows:

Theorem 1. *Suppose that for the MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$ there exist MSPs $\mathcal{M}_3$ and $\mathcal{M}_4$ such that $\mathcal{M}_1 \diamond \mathcal{M}_2 = \mathcal{M} = \mathcal{M}_3 \diamond \mathcal{M}_4$. Then the sufficient condition for existence of general perfect information-theoretically secure MPC secure against $(\Delta_1, \Delta_2, \Delta_A)$-adversary is*

$$\Gamma_A^{\perp} \subseteq \Gamma \subseteq \Gamma_1 \uplus \Gamma_2, \quad (\Gamma_A \uplus \Gamma_A)^{\perp} \subseteq \Gamma_i \quad \textit{for} \quad i = 1, 2, 3,$$

where Γ is the access structure computed by the strongly multiplicative resulting MSP $\mathcal{M}$ from MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$ and/or from MSPs $\mathcal{M}_3$ and $\mathcal{M}_4$.

We will call the access structure Γ_3 (the MSP $\mathcal{M}_3$, resp.) "reduced". It is easy to see that such MSPs $\mathcal{M}_3$ and $\mathcal{M}_4$ always exist, e.g. $\mathcal{M}_1 = \mathcal{M}_3$ and $\mathcal{M}_2 = \mathcal{M}_4$. In the threshold case there exist several pairs of MSPs that satisfy the assumption of Theorem 1.
Note also that the Maurer's [14] necessary and sufficient condition $P \notin \Delta_1 \uplus \Delta_1 \uplus \Delta_A$ is satisfied (in case $\Gamma_1 = \Gamma_2$), on the other hand this conditions does not guarantee that $\Gamma_A^{\perp} \subseteq \Gamma$, when $\Gamma \neq \Gamma_1 \uplus \Gamma_2$, i.e., $\Gamma \subset \Gamma_1 \uplus \Gamma_2$.
The picture in the general access structure appears to be analogous to this in the threshold case [7,9]. Remarkably the conditions in the information-theoretic settings are "similar" to the conditions in the cryptographic settings (see the result of Nikov *et al.* for the computational model). Note that it is not required anymore Γ to satisfy the VSS conditions.
If we compare with the protocol in [4] we can see that now the player who re-shares his share do not need to commit to every single entry in the used vector. Hence the number of the used VSS is reduced. Also note that this protocol does not depend on the model considered here (Nikov *et al.*), it could be applied also for the model of Cramer *et al.*
The paper is organized as follows: In Section 2 the information-theoretically secure VSS, randomization and re-sharing protocols are presented. In Section 3 we introduce some terminology and concepts, we state the results and explain the role they play in comparison with earlier results.

2 Background

2.1 VSS – Share Phase

Let the dealer $\mathcal{D}$ shares the secret s to the players P_i using the VSS protocol, as described by Cramer *et al.* in [4], and let $\mathcal{M}$ be an MSP with matrix M $(m \times d)$.

1. The Dealer $\mathcal{D}$ chooses a symmetric $d \times d$ matrix R subject to s (the secret) in its upper left corner.
2. The Dealer $\mathcal{D}$ gives to the participant P_i shares $v_{\varphi(i)} = M_{\varphi(i)}R$ ($v_{\varphi(i)}$ is $|\varphi(i)| \times d$ matrix), where the "true part" (which will be used in the reconstruction) of the shares is $v_{\varphi(i)}\varepsilon$.
3. The players P_i and P_j perform a pairwise-check as follows:

$$M_{\varphi(j)}v_{\varphi(i)}^T = M_{\varphi(j)}RM_{\varphi(i)}^T = v_{\varphi(j)}M_{\varphi(i)}^T.$$

2.2 VSS – Reconstruction Phase

For any group of players $G \in \Gamma$ there exists a recombination vector $\lambda_{\varphi(G)}$, such that they can reconstruct together the secret s as follows:

$$(v_{\varphi(G)}\varepsilon)\lambda_{\varphi(G)}^T = \langle \lambda_{\varphi(G)}, v_{\varphi(G)}\varepsilon \rangle = \sum_{i \in G} \lambda_{\varphi(i)}(v_{\varphi(i)}\varepsilon) = s.$$

2.3 Information-Theoretic Homomorphic Commitments and Re-share Phase

In the re-share phase each player P_i plays the role of the dealer sharing the true part of his shares among the participants using VSS with the same MSP $\mathcal{M}$.

1. Any player P_i re-shares his true part of the share $v_{\varphi(i)}\varepsilon$, i.e., for any $i_1 \in \varphi(i)$ he chooses a symmetric $d \times d$ matrix $R^{(i_1)}$ such that its first row (column) is v_{i_1} and the value in its upper left corner is $v_{i_1}\varepsilon$.
2. P_i sends to P_j temporary shares $y_{i_1,\varphi(j)} = M_{\varphi(j)}R^{(i_1)}$, whose true part is $y_{i_1,\varphi(j)}\varepsilon$.
3. The players P_k and P_j perform the usual commitment verification (VSS pairwise-check):

$$M_{\varphi(j)}y^T_{i_1,\varphi(k)} = M_{\varphi(j)}R^{(i_1)}M^T_{\varphi(k)} = y_{i_1,\varphi(j)}M^T_{\varphi(k)}.$$

4. In addition P_j checks his true part of the share

$$y_{i_1,\varphi(j)}\varepsilon = M_{\varphi(j)}R^{(i_1)}\varepsilon = M_{\varphi(j)}v^T_{i_1} = v_{\varphi(j)}M^T_{i_1}.$$

   The last equality is the pair-wise check in VSS (step 3 in the Share phase). Note that this additional check ensures that the player P_i really re-shares his share, i.e., he is honest.
5. As usual for any group of players $\widetilde{G} \in \Gamma$ there exists a recombination vector $\widetilde{\lambda}_{\varphi(\widetilde{G})}$ such that they can together reconstruct the true part of the initial share – $v_{i_1}\varepsilon$.

$$(y_{i_1,\varphi(\widetilde{G})}\varepsilon)\widetilde{\lambda}^T_{\varphi(\widetilde{G})} = \langle\widetilde{\lambda}_{\varphi(\widetilde{G})}, y_{i_1,\varphi(\widetilde{G})}\varepsilon\rangle = \sum_{j\in\widetilde{G}}\widetilde{\lambda}_{\varphi(j)}(y_{i_1,\varphi(j)}\varepsilon) = v_{i_1}\varepsilon.$$

6. Denote the list of good players by $\mathcal{L} \in \Gamma$. Then P_j, using the corresponding recombination vector $\lambda_{\varphi(\mathcal{L})}$, computes

$$z_{\varphi(j)} = \sum_{i\in\mathcal{L}}\lambda_{\varphi(i)}y_{\varphi(i),\varphi(j)}.$$

The new shares (of the same secret s) are $z_{\varphi(j)}$ and they satisfy all the necessary properties as follows:

- The pair-wise check holds:

$$\begin{aligned} M_{\varphi(k)}z^T_{\varphi(j)} &= \sum_{i\in\mathcal{L}}\lambda_{\varphi(i)}M_{\varphi(k)}y^T_{\varphi(i),\varphi(j)} \\ &= (\sum_{i\in\mathcal{L}}\lambda_{\varphi(i)}y_{\varphi(i),\varphi(k)})M^T_{\varphi(j)} = z_{\varphi(k)}M^T_{\varphi(j)}. \end{aligned}$$

- The players in any group $\widetilde{G} \in \Gamma$ can reconstruct the secret s together.

$$\begin{aligned}(z_{\varphi(\widetilde{G})}\varepsilon)\widetilde{\lambda}^T_{\varphi(\widetilde{G})} &= \langle \widetilde{\lambda}_{\varphi(\widetilde{G})}, z_{\varphi(\widetilde{G})}\varepsilon\rangle = \sum_{j\in\widetilde{G}} \widetilde{\lambda}_{\varphi(j)}(z_{\varphi(j)}\varepsilon) \\ &= \sum_{j\in\widetilde{G}} \widetilde{\lambda}_{\varphi(j)}(\sum_{i\in\mathcal{L}} \lambda_{\varphi(i)}(y_{\varphi(i),\varphi(j)}\varepsilon)) \\ &= \sum_{i\in\mathcal{L}} \lambda_{\varphi(i)}(\sum_{j\in\widetilde{G}} \widetilde{\lambda}_{\varphi(j)}(y_{\varphi(i),\varphi(j)}\varepsilon)) = \sum_{i\in\mathcal{L}} \lambda_{\varphi(i)}(v_{\varphi(i)}\varepsilon) = s.\end{aligned}$$

2.4 The Randomization Phase

We can use the Renewal phase from [16] as a randomization protocol.

3 Reduction Protocol

3.1 The Set-up

Let Γ_1 and Γ_2 be access structures, computed by MSPs $\mathcal{M}_1 = (\mathbb{F}, M_1, \varepsilon_1, \psi_1)$ and $\mathcal{M}_2 = (\mathbb{F}, M_2, \varepsilon_2, \psi_2)$, respectively. Let also M_1 be $m_1 \times d_1$ matrix, M_2 be $m_2 \times d_2$ matrix and φ_1, φ_2 be the "inverse" functions of ψ_1 and ψ_2.
Let $\mathcal{M} = \mathcal{M}_1 \diamond \mathcal{M}_2$ be the multiplicative resulting MSP, i.e., $\mathcal{M} = (\mathbb{F}, M = M_1 \diamond M_2, \varepsilon = \varepsilon_1 \diamond \varepsilon_2, \psi)$, where $\psi(i,j) = r$ if and only if $\psi_1(i) = \psi_2(j) = r$. Hence M is $m \times d_1 d_2$ matrix, where $m = \sum_i |\varphi_1(i)||\varphi_2(i)| = \sum_i |\varphi(i)|$. Let us consider the access structure Γ computed by the MSP $\mathcal{M}$.
Let the first secret s_1 is shared using VSS by MSP $\mathcal{M}_1$ with symmetric $d_1 \times d_1$ matrix $R^{(1)}$, i.e., $v_{\varphi_1(i)} = (M_1)_{\varphi_1(i)}R^{(1)}$ be the shares of P_i ($v_{\varphi_1(i)}$ is $|\varphi_1(i)| \times d_1$ matrix). The "true part" of the shares are the first coordinates of each share, i.e., $v_{\varphi_1(i)}\varepsilon_1$.
Analogously, let the second secret s_2 is shared by MSP $\mathcal{M}_2$ with symmetric $d_2 \times d_2$ matrix $R^{(2)}$, i.e., $w_{\varphi_2(i)} = (M_2)_{\varphi_2(i)}R^{(2)}$ be the shares of P_i. ($w_{\varphi_2(i)}$ is $|\varphi_2(i)| \times d_2$ matrix). The "true part" of the shares are the first coordinates of each share, i.e., $w_{\varphi_2(i)}\varepsilon_2$.

3.2 Local Computation Phase

Denote by $R = R^{(1)} \otimes R^{(2)}$ a $d_1 d_2 \times d_1 d_2$ symmetric matrix. Note that the value in the upper left corner of R is the product $s_1 s_2$. Let us choose the indices $i_1 \in \varphi_1(i)$, $i_2 \in \varphi_2(i)$, $j_1 \in \varphi_1(j)$ and $j_2 \in \varphi_2(j)$.
If the player P_i locally computes $\otimes$ product of his shares he obtains his new shares $v_{\varphi_1(i)} \otimes w_{\varphi_2(i)}$ (which are an $|\varphi(i)| \times d_1 d_2$ matrix).
This shares correspond to an MSP $\mathcal{M}$ and the random matrix R as defined above, i.e., $((M_1)_{i_1} \otimes (M_2)_{i_2})R = v_{i_1} \otimes w_{i_2}$.

The pair-wise check for the new shares also holds:

$$((M_1)_{i_1} \otimes (M_2)_{i_2})(v_{j_1} \otimes w_{j_2})^T = ((M_1)_{i_1} v_{j_1}^T)((M_2)_{i_2} w_{j_2}^T) = \\ (v_{i_1}(M_1)_{j_1}^T)(w_{i_2}(M_2)_{j_2}^T) = (v_{i_1} \otimes w_{i_2})((M_1)_{j_1} \otimes (M_2)_{j_2})^T.$$

Note that the new "true part" of the shares is the product

$$(v_{\varphi_1(i)} \otimes w_{\varphi_2(i)})\varepsilon = (v_{\varphi_1(i)}\varepsilon_1) \otimes (w_{\varphi_2(i)}\varepsilon_2).$$

In the new MSP $\mathcal{M}$ for any group of players $G \in \Gamma$ there exists a recombination vector $\lambda_{\varphi(G)}$ such that they can reconstruct together the product of the secrets – $s_1 s_2$.

$$((v_{\varphi_1(G)} \otimes w_{\varphi_2(G)})\varepsilon)\lambda_{\varphi(G)}^T = \langle \lambda_{\varphi(G)}, (v_{\varphi_1(G)} \otimes w_{\varphi_2(G)})\varepsilon \rangle \\ = \sum_{j \in G} \lambda_{\varphi(j)}((v_{\varphi_1(j)} \otimes w_{\varphi_2(j)})\varepsilon) = s_1 s_2.$$

3.3 Decomposition – Change of the Basis

Let d_3 and d_4 are integers such that $d_1 d_2 = d_3 d_4$ and, as usual, $\varepsilon_3 \in \mathbb{F}^{d_3}$ be the unit column vector. Denote by $e_i = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{F}^{d_4}$ the unit row vectors, for $i = 1, \ldots, d_4$.
It is easy to see that there exist uniquely defined vectors $x_{j_1,j_2}^{(i)}, \widetilde{x}_{j_1,j_2}^{(i)} \in \mathbb{F}^{d_3}$ for $i = 1, \ldots, d_4$, such that the following equalities hold

$$v_{j_1} \otimes w_{j_2} = \sum_{i=1}^{d_4} x_{j_1,j_2}^{(i)} \otimes e_i; \qquad v_{j_1} \otimes w_{j_2} = \sum_{i=1}^{d_4} e_i \otimes \widetilde{x}_{j_1,j_2}^{(i)}. \tag{2}$$

Note that $(v_{j_1} \otimes w_{j_2})\varepsilon = x_{j_1,j_2}^{(1)}\varepsilon_3 = \widetilde{x}_{j_1,j_2}^{(1)}\varepsilon_3$.

3.4 Degree Reduction Phase

Let Γ_3 be an access structure, computed by the MSP $\mathcal{M}_3 = (\mathbb{F}, M_3, \varepsilon_3, \psi_3)$. Let also M_3 be $m_3 \times d_3$ matrix and φ_3 be the "inverse" functions of ψ_3.
Any player P_j re-shares the first coordinate of the vector $x_{j_1,j_2}^{(i)}$, i.e., $x_{j_1,j_2}^{(i)}\varepsilon_3$ for $i = 1, \ldots, d_4$ using VSS Share protocol. Let us denote the different copies of VSSs by $VSS(i)$. For each VSS the player uses a symmetric $d_3 \times d_3$ matrix $R_{j_1,j_2}^{(i)}$, such that its first row (column) is $x_{j_1,j_2}^{(i)}$. So, the player P_k receives from P_j the following temporary shares:

$$y_{j_1,j_2,\varphi_3(k)}^{(i)} = (M_3)_{\varphi_3(k)} R_{j_1,j_2}^{(i)}$$

As in Subsection 2.3 the player P_k verifies the commitments of P_j using usual pair-wise check for each $VSS(i)$.

3.5 Additional Check on the Degree Reduction Phase

Now we need to ensure that the player P_j re-shares the correct vectors $x^{(i)}_{j_1,j_2}$ and in particular their true part. Unfortunately we can not apply directly the additional check procedure from step 4. in the re-share protocol, because in the degree reduction phase we use two different access structures.
Let us choose the indices $j_1 \in \varphi_1(j)$, $j_2 \in \varphi_2(j)$, $k_1 \in \varphi_1(k)$, $k_2 \in \varphi_2(k)$, $k_3 \in \varphi_3(k)$ and $k_4 \in \varphi_4(k)$. In order to perform this additional check we assume that there exist matrices M_3 and M_4, such that $M_1 \diamond M_2 = M = M_3 \diamond M_4$. This assumption means that we have $(M_3)_{k_3} \otimes (M_4)_{k_4} = (M_1)_{k_1} \otimes (M_2)_{k_2}$ for some rows k_1, k_2, k_3, k_4 of the corresponding matrices.
We first prove the following three equalities.

$$\begin{aligned}\langle y^{(i)}_{j_1,j_2,k_3}, \varepsilon_3^T\rangle &= \langle (M_3)_{k_3} R^{(i)}_{j_1,j_2}, \varepsilon_3^T\rangle \\ &= \langle (M_3)_{k_3}, (R^{(i)}_{j_1,j_2})_1\rangle = \langle (M_3)_{k_3}, x^{(i)}_{j_1,j_2}\rangle,\end{aligned} \tag{3}$$

$$\langle (M_3)_{k_3} \otimes (M_4)_{k_4}, x^{(i)}_{j_1,j_2} \otimes e_i\rangle = \langle (M_3)_{k_3}, x^{(i)}_{j_1,j_2}\rangle \langle (M_4)_{k_4}, e_i\rangle, \tag{4}$$

$$\begin{aligned}&\langle (M_1)_{k_1} \otimes (M_2)_{k_2}, v_{j_1} \otimes w_{j_2}\rangle = \langle (M_1)_{k_1}, v_{j_1}\rangle \langle (M_2)_{k_2}, w_{j_2}\rangle \\ &= ((M_1)_{k_1} v_{j_1}^T)((M_2)_{k_2} w_{j_2}^T) = (v_{k_1}(M_1)_{j_1}^T)(w_{k_2}(M_2)_{j_2}^T) \\ &= \langle (M_1)_{j_1}, v_{k_1}\rangle \langle (M_2)_{j_2}, w_{k_2}\rangle.\end{aligned} \tag{5}$$

Now using (2) together with (3),(4), and (5) we are ready to prove that the player P_k can make an additional check whether P_j re-shared correctly the shares in the degree reduction phase. To perform this check P_k uses his old shares v_{k_1} and w_{k_2} together with the newly received shares $y^{(i)}_{j_1,j_2,k_3}$ from P_j and some public information.

$$\langle (M_1)_{j_1}, v_{k_1}\rangle \langle (M_2)_{j_2}, w_{k_2}\rangle = \sum_{i=1}^{d_4} \langle (M_4)_{k_4}, e_i\rangle \langle y^{(i)}_{j_1,j_2,k_3}, \varepsilon_3^T\rangle.$$

Note that we can simply choose $\mathcal{M}_3 = \mathcal{M}_1$ and $\mathcal{M}_4 = \mathcal{M}_2$, in this case we have $\Gamma_1 = \Gamma_3$.

3.6 The New Shares

Finally, in order to complete the protocol we need to define the new shares. Recall that $j_1 \in \varphi_1(j)$ and $j_2 \in \varphi_2(j)$ if and only if $\{j_1, j_2\} \in \varphi(j)$. That is way we will denote $x^{(i)}_{j_1,j_2}$ and $y^{(i)}_{j_1,j_2,\varphi_3(k)}$ for $j_1 \in \varphi_1(j)$ and $j_2 \in \varphi_2(j)$ also by $x^{(i)}_{\varphi(j)}$ and by $y^{(i)}_{\varphi(j),\varphi_3(k)}$.
As we mentioned earlier in Section 3.4 for any group of players $\widetilde{G} \in \Gamma_3$ there exists a recombination vector $\widetilde{\lambda}_{\varphi_3(\widetilde{G})}$ such that they can reconstruct together the

first coordinate of the vector $x^{(i)}_{\varphi(j)}$, i.e., $x^{(i)}_{\varphi(j)}\varepsilon_3$, for $i = 1, \ldots, d_4$ (reconstruction phase of $VSS(i)$) as follows:

$$(y^{(i)}_{\varphi(j),\varphi_3(\widetilde{G})}\varepsilon_3)\widetilde{\lambda}^T_{\varphi_3(\widetilde{G})} = \langle \widetilde{\lambda}_{\varphi_3(\widetilde{G})}, y^{(i)}_{\varphi(j),\varphi_3(\widetilde{G})}\varepsilon_3\rangle = \sum_{k\in\widetilde{G}} \widetilde{\lambda}_{\varphi_3(k)}(y^{(i)}_{\varphi(j),\varphi_3(k)}\varepsilon_3) = x^{(i)}_{\varphi(j)}\varepsilon_3, \tag{6}$$

Note also that for any group of players $G \in \Gamma$ there exists a recombination vector $\lambda_{\varphi(G)}$ such that they can reconstruct together the product of the secrets $s_1 s_2$.

$$(x^{(1)}_{\varphi(G)}\varepsilon_3)\lambda^T_{\varphi(G)} = \langle \lambda_{\varphi(G)}, x^{(1)}_{\varphi(G)}\varepsilon_3\rangle = \langle \lambda_{\varphi(G)}, (v_{\varphi_1(G)} \otimes w_{\varphi_2(G)})\varepsilon\rangle = s_1 s_2. \tag{7}$$

(Here the last equality from Subsection 3.2 and the note from Subsection 3.3 are used.)

Now we are ready to define the new shares. Denote the list of good players by $\mathcal{L} \in \Gamma$, then P_k computes his new shares as follows:

$$z_{\varphi_3(k)} = \sum_{j\in\mathcal{L}} \lambda_{\varphi(j)} y^{(1)}_{\varphi(j),\varphi_3(k)}.$$

For the new shares $z_{\varphi_3(k)}$ the pair-wise check holds:

$$(M_3)_{\varphi_3(i)} z^T_{\varphi_3(k)} = \sum_{j\in\mathcal{L}} \lambda_{\varphi(j)} (M_3)_{\varphi_3(i)} (y^{(1)}_{\varphi(j),\varphi_3(k)})^T = (\sum_{i\in\mathcal{L}} \lambda_{\varphi(j)} y^{(1)}_{\varphi(j),\varphi_3(i)})(M_3)^T_{\varphi_3(k)} = z_{\varphi_3(i)}(M_3)^T_{\varphi_3(k)}.$$

For any $\widetilde{G} \in \Gamma_3$ the players can reconstruct together the product $s_1 s_2$ using (6) and (7) as follows:

$$\begin{aligned}(z_{\varphi_3(\widetilde{G})}\varepsilon_3)\widetilde{\lambda}^T_{\varphi_3(\widetilde{G})} &= \langle \widetilde{\lambda}_{\varphi_3(\widetilde{G})}, z_{\varphi_3(\widetilde{G})}\varepsilon_3\rangle = \sum_{k\in\widetilde{G}} \widetilde{\lambda}_{\varphi_3(k)}(z_{\varphi_3(k)}\varepsilon_3) \\ &= \sum_{k\in\widetilde{G}} \widetilde{\lambda}_{\varphi_3(k)}(\sum_{j\in\mathcal{L}} \lambda_{\varphi(j)}(y^{(1)}_{\varphi(j),\varphi_3(k)}\varepsilon_3)) \\ &= \sum_{j\in\mathcal{L}} \lambda_{\varphi(j)}(\sum_{k\in\widetilde{G}} \widetilde{\lambda}_{\varphi_3(k)}(y^{(1)}_{\varphi(j),\varphi_3(k)}\varepsilon_3)) \\ &= \sum_{j\in\mathcal{L}} \lambda_{\varphi(j)}(x^{(1)}_{\varphi(j)}\varepsilon_3) = s_1 s_2\end{aligned}$$

At the end of the protocol each player P_k possesses new shares $z_{\varphi_3(k)}$ of MSP $\mathcal{M}_3$ (computing the access structure Γ_3) of the product $s_1 s_2$.

Lemma 1. *Suppose that for the MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$ there exist MSPs $\mathcal{M}_3$ and $\mathcal{M}_4$ such that*

$$\mathcal{M}_1 \diamond \mathcal{M}_2 = \mathcal{M} = \mathcal{M}_3 \diamond \mathcal{M}_4.$$

Let Γ be the access structure computed by the strongly multiplicative resulting MSP $\mathcal{M}$ from MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$ and/or from MSPs $\mathcal{M}_3$ and $\mathcal{M}_4$ and let also the access structures Γ and Γ_i for $i = 1, 2, 3$ satisfy the conditions

$$\Gamma_A^{\perp} \subseteq \Gamma \subseteq \Gamma_1 \uplus \Gamma_2, \quad (\Gamma_A \uplus \Gamma_A)^{\perp} \subseteq \Gamma_i \quad \text{for} \quad i = 1, 2, 3.$$

Then the "degree reduction" protocol is information-theoretically secure against $(\Delta_1, \Delta_2, \Delta_A)$-adversary.

Due to lack of space we will not give a formal security proof for our protocol. However, to have a feeling why it is secure, note first that in the re-sharing phase every player could verify whether the "true" part of his share is correct or not. Then, as in the protocol from [4], the shares of the players (in our case the "true" part of the shares) have to satisfy a fixed linear relation, which allow every player to complain against incorrect re-sharing.

3.7 Complexity Issues

In this subsection we will follow [4]. Define $msp_{\mathbb{F}}(f)$ to be the size of the smallest MSP over $\mathbb{F}$ computing a monotone boolean function f. Next define $\mu_{\mathbb{F}}(f)$ to be the size of the smallest multiplicative MSP over $\mathbb{F}$ computing f. Similarly, $\mu^*_{\mathbb{F}}(f)$ to be the size of the smallest strongly multiplicative MSP. In other words for a given adversary A with adversary structure Δ_A we require for every set $B \in \Delta_A$ to have $B \notin \Gamma$, but $B^c \in \Gamma$. By definition, we have $msp_{\mathbb{F}}(f) \leq \mu_{\mathbb{F}}(f) \leq \mu^*_{\mathbb{F}}(f)$. In [4] Cramer *et al.* characterized the functions that (strongly) multiplicative MSP's can compute, and proved that the multiplication property for an MSP can be assumed without loss of efficiency. In particular, for the passive (multiplicative) case they proved that $\mu_{\mathbb{F}}(f) \leq 2\ msp_{\mathbb{F}}(f)$ provided that f is Q^2 function. Unfortunately there is no similar result for the strongly multiplicative case. Instead the authors in [4] proved that for an active adversary $\mu^*_{\mathbb{F}}(f)$ is bounded by the so-called "formula complexity".
In the recent paper of Nikov *et al.* [17] a different approach is considered. Recall that in that model given an Q^3 adversary A we are looking for two access structures (resp. monotone boolean functions) Γ_1 and Γ_2 (resp. f_1 and f_2) such that their strongly multiplicative resulting MSP computes Γ (resp. f). Or in other words for a given adversary A with adversary structure Δ_A we require that for every set $B \in \Delta_A$ to have $B \notin \Gamma_1$, $B \notin \Gamma_2$ but $B^c \in \Gamma$. Let us define $\nu_{\mathbb{F}}(f)$ to be the size of the smallest strongly multiplicative resulting MSP over $\mathbb{F}$ computing f. How these two measures $\mu^*_{\mathbb{F}}(f)$ and $\nu_{\mathbb{F}}(f)$ are related as well as whether this new notion give us better measure for the complexity of an MPC is subject of ongoing research.

Acknowledgements. The authors would like to thank Ronald Cramer for the careful reading of earlier versions of the paper and for his constructive comments and remarks.

References

1. M. Ben-Or, S. Goldwasser and A. Wigderson, Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation, *Proc. ACM STOC'88*, 1988, pp. 1–10.
2. D. Chaum, C. Crepeau and I. Damgard, Multi-Party Unconditionally Secure Protocols, *Proc. ACM STOC'88*, 1988, pp. 11–19.
3. R. Cramer, Introduction to Secure Computation, *Lectures on Data Security – Modern Cryptology in Theory and Practice*, Springer-Verlag LNCS 1561, 1999, pp. 16–62.
4. R. Cramer, I. Damgard and U. Maurer, General Secure Multi-Party Computation from any Linear Secret Sharing Scheme, *Proc. EUROCRYPT 2000*, Springer-Verlag LNCS 1807, 2000, pp. 316–334.
5. R. Cramer, S. Fehr, Optimal Black-Box Secret Sharing over Arbitrary Abelian Groups, *Proc. CRYPTO 2002*, Springer-Verlag LNCS 2442, 2002, pp. 272–287.
6. B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults, *Proc. of the IEEE 26th Annual Symp. on Foundations of Computer Science*, 1985, pp. 383–395.
7. I.Damgard, An Error in the Mixed Adversary Protocol by Fitzi, Hirt and Maurer, *Bricks Report*, RS-99-2, 1999.
8. S. Fehr, U. Maurer, Linear VSS and Distributed Commitments Based on Secret Sharing and Pairwise Checks, *Proc. CRYPTO 2002*, Springer Verlag LNCS 2442, 2002, pp. 565–580.
9. M.Fitzi, M.Hirt and U.Maurer, Trading Correctness for Privacy in Unconditional Multi-Party Computation, *Proc. CRYPT0'98*, Springer-Verlag, LNCS 1462, 1998, pp. 121–136.
10. R. Gennaro, M. Rabin, T. Rabin, Simplified VSS and Fast-Track Multi-party Computations with Applications to Threshold Cryptography, *Proc. ACM PODC'98*, 1998.
11. O. Goldreich, S. Micali and A. Wigderson, How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority, *Proc. ACM STOC'87*, 1987, pp. 218–229.
12. M. Hirt, U. Maurer, Player Simulation and General Adversary Structures in Perfect Multi-party Computation, *J. of Cryptology* 13, 2000, pp. 31–60.
13. M. Karchmer, A. Wigderson, On Span Programs, *Proc. of 8-th Annual Structure in Complexity Theory Conference*, San Diego, California, 18–21 May 1993. IEEE Computer Society Press, pp. 102–111.
14. U. Maurer, Secure Multi-Party Computation Made Simple, *3rd Conference on Security in Communication Networks*, September 12–13, 2002, Amalfi, Italy, Springer-Verlag LNCS 2576, 2003, pp. 14–28.
15. V. Nikov, S. Nikova, B. Preneel, J. Vandewalle, Applying General Access Structure to Proactive Secret Sharing Schemes, *Proc. of the 23rd Symposium on Information Theory in the Benelux*, May 29–31, 2002, Universite Catolique de Lovain (UCL), Lovain-la-Neuve, Belgium, pp. 197–206, *Cryptology ePrint Archive*: Report 2002/141.

16. V. Nikov, S. Nikova, B. Preneel, J. Vandewalle, On Distributed Key Distribution Centers and Unconditionally Secure Proactive Verifiable Secret Sharing Schemes based on General Access Structure, *INDOCRYPT* 2002, Springer-Verlag LNCS 2551, 2002, pp. 422–437.
17. V. Nikov, S. Nikova, B. Preneel, Multi-Party Computation from any Linear Secret Sharing Scheme Secure against Adaptive Adversary: The Zero-Error Case, *Cryptology ePrint Archive*: Report 2003/006.
18. A. Shamir, How to share a secret, *Commun. ACM* 22, 1979, pp. 612–613.

Optimized χ^2-Attack against RC6

Norihisa Isogai*, Takashi Matsunaka, and Atsuko Miyaji

Japan Advanced Institute of Science and Technology.
{isogai, t-matsuna, miyaji }@jaist.ac.jp

Abstract. In this paper, we make progress on χ^2-attack by introducing the optimization. We propose three key recovery attacks against RC6 without post-whitening, and apply these three key recovery algorithms to RC6. We discuss their differences and optimization and thus our best attack can break 16-round RC6 without pre-whitening with 128-bit key (resp. 16-round RC6 with 192-bit key) by using $2^{117.84}$ (resp. $2^{122.84}$) chosen plaintexts with a success probability of 95% (resp. 90%). As far as the authors know, this is the best result of attacks to RC6.

Keywords: Block Cipher, Cryptanalysis, RC6, χ^2-attack

1 Introduction

RC6 [11] is a block cipher designed by Rivest *et al.* in 1998. RC6-$w/r/b$ means that four w-bit-word plaintexts are encrypted with r rounds by b-byte keys. Currently, RC6-32/20 is recommended to give sufficient resistance against the known attacks [1,2,3,5,7,9,12,13]. In this paper, RC6-32 is simply denoted by RC6. RC6 operates as an unit of w-bit word using five basic operations such as an addition, a subtraction, a bitwise exclusive-or, a multiplication, and a data dependent rotation. Therefore, this block cipher has a wonderful capability for performing high-speed software implementation especially on Intel processors. Up to the present, linear attacks, differential attacks, and χ^2-attacks against RC6 and some simplified variants of RC6 have been analyzed intensively. Table 1 summarizes the previous results on RC6. In [2], the security of RC6 against the differential and linear cryptanalysis was given. They estimated that 12-round RC6 is not secure against the differential cryptanalysis. As for linear cryptanalysis using multiple approximations and linear hulls, it was reported that RC6 with 16 or more rounds is secure. As a result, they concluded that 20-round RC6 is secure against differential and linear cryptanalysis. In [12], on the other hand, a correct key of 14-round RC6 with 256-bit key can be recovered by using multiple linear attack, and a weak key of 18-round RC6 can be recovered with the probability of about $1/2^{90}$.

The χ^2-attack is one of the most effective attacks on RC6. The χ^2-attack was originally proposed by Vaudenay as an attack on the Data Encryption Standard

* The author is currently with Matsushita Information System Research Laboratory Nagoya Co., LTD.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 16–32, 2003.

(DES) [14], and Handschuh *et al.* applied that to SEAL [6]. In [5,7,9], the χ^2-attacks were applied to RC6 or a simplified variant of RC6. The χ^2-attack can be used for both distinguishing attacks and key recovery attacks. Distinguishing attacks handle plaintexts in such a way that the χ^2-value of a part of ciphertexts becomes significantly a higher value. Key recovery attacks have to rule out all wrong keys, single out exactly a correct key by using the χ^2-value, and thus they often require more work and memory than distinguishing attacks. In [5,7], they just focused on such plaintexts that outputs high χ^2-value on ciphertext, and in [9], they made progress by introducing a notion of *variance* as well as χ^2-value itself. But, unfortunately, optimization of χ^2-value has never been discussed, that is, what level of variance is optimal.

In this paper, we propose three key recovery attacks against RC6 without post-whitening and discuss the differences and optimization. We also apply the key recovery attacks to RC6 and demonstrate one of them on RC6-8. Our key recovery attack itself gives a remarkable impact on RC6: our best attack can break 16-round RC6 without pre-whitening with 128-bit key (resp. 16-round RC6 with 192-bit key) by using $2^{117.84}$ (resp. $2^{122.84}$) chosen plaintexts with a success probability of 95% (resp. 90%).

This paper is organized as follows. Section 2 summarizes the notation, RC6 algorithm, and the χ^2-test. Section 3 investigates the χ^2-statistic of RC6. Section 4 presents three key recovery attacks against RC6 without post-whitening, Algorithms 2, 3, and 4. We evaluate the security against RC6 in Section 5. Conclusion is given in Section 6.

Table 1. Attacks on RC6

Attack	Target RC6	Rounds	#Texts
Linear Attack [2]	RC6	16	2^{119}
χ^2 Attack [7]	RC6 with 256-bit key	15	2^{119}
Multiple Linear Attack [12]	RC6 with 192-bit key	14[1]	$2^{119.68}$
χ^2 Attack [9]	RC6W[2] with 128-bit key	17	$2^{123.9}$
Our result	RC6P[3] with 128-bit key	16	$2^{117.84}$
	RC6 with 192-bit key	16	$2^{122.84}$

1: A weak key of 18-round RC6 with 256-bit key can be recovered by $2^{126.936}$ plaintexts with the probability of about $1/2^{90}$.
2: RC6W means RC6 without pre- or post-whitening.
3: RC6P means RC6 without post-whitening.

2 Preliminary

We summarize the notation, RC6 algorithm, and the χ^2-test, used in this paper.

2.1 Notation

Let us use the following notation:

$+$: addition;
$-$: subtraction;

$\oplus$: bitwise exclusive-or;
r : number of rounds;
$a \lll b$: cyclic rotation of a to the left by b-bit;
$a \ggg b$: cyclic rotation of a to the right by b-bit;
(A_i, B_i, C_i, D_i) : input of the i-th round; (A_0, B_0, C_0, D_0) : plaintext;
$(A_{r+2}, B_{r+2}, C_{r+2}, D_{r+2})$: ciphertext after r-round encryption;
S_i : i-th subkey;
$\mathrm{lsb}_n(X)$: least significant n-bit of X;
$\mathrm{msb}_n(X)$: most significant n-bit of X;
X^i : i-th bit of X;
$f(x)$: $x \times (2x + 1)$;
$F(x)$: $f(x) \pmod{2^{32}} \lll 5$;
$x||y$: concatenated value of x and y.

We denote the least significant bit (lsb) to the 1st bit, and the most significant bit (msb) as the 32-th bit for any 32-bit element.

2.2 Block Cipher RC6

Algorithm 1 (RC6 Encryption Algorithm)

1. $A_1 = A_0$; $B_1 = B_0 + S_0$; $C_1 = C_0$; $D_1 = D_0 + S_1$;
2. for $i = 1$ to r do: $t = F(B_i)$; $u = F(D_i)$;
   $A_{i+1} = B_i$; $B_{i+1} = ((C_i \oplus u) \lll t) + S_{2i+1}$; $C_{i+1} = D_i$;
   $D_{i+1} = ((A_i \oplus t) \lll u) + S_{2i}$;
3. $A_{r+2} = A_{r+1} + S_{2r+2}$; $B_{r+2} = B_{r+1}$; $C_{r+2} = C_{r+1} + S_{2r+3}$; $D_{r+2} = D_{r+1}$.

Parts 1 and 3 of Algorithm 1 are called pre-whitening and post-whitening, respectively. We call the version of RC6 without post-whitening to, simply, RC6P.

2.3 χ^2-Test

We make use of the χ^2-tests for distinguishing a non-uniformly random distribution from uniformly random distribution [7]. Let $X = X_0, ..., X_{n-1}$ be a sequence with $\forall X_i \in \{a_0, \cdots, a_{m-1}\}$. Let $N_{a_j}(X)$ be the number of X_i which equals a_j. The χ^2-statistic of X, $\chi^2(X)$, estimates the difference between X and the uniform distribution as follows:

$$\chi^2(X) = \frac{m}{n} \sum_{i=0}^{m-1} \left(N_{a_i}(X) - \frac{n}{m} \right)^2 .$$

Table 2 presents each threshold for 63 degrees of freedom. For example, (level, χ^2) = (0.95, 82.53) for 63 degrees of freedom in Table 2 means that the value of χ^2-statistic exceeds 82.53 in the probability of 5% if the observation X is uniform.

3 χ^2-Statistic of RC6

We improve the distinguishing attacks in such a way that the χ^2-values become significantly high and that the available number of plaintexts is not reduced.

Table 2. Selected threshold values of χ^2-distribution with 63 degrees of freedom

Level	0.50	0.60	0.70	0.80	0.90	0.95	0.99
63 degrees of freedom	62.33	65.20	68.37	72.20	77.75	82.53	92.01

3.1 Overview of Experiments

The previous results [7,9] of χ^2-statistic are summarized as follows: 1. 10-bit outputs of $\mathrm{lsb}_5(A_{r+1})||\mathrm{lsb}_5(C_{r+1})$ lead to much stronger biases if both $\mathrm{lsb}_5(A_0)$ and $\mathrm{lsb}_5(C_0)$ are fixed and both B_0 and D_0 introduce zero rotation in the 1st round;
2. 10-bit outputs of $\mathrm{lsb}_5(A_{r+1})||\mathrm{lsb}_5(C_{r+1})$ lead to much stronger biases if $\mathrm{lsb}_5(A_0)$ is fixed, $\mathrm{lsb}_5(C_0) = 0$, and both B_0 and D_0 introduce zero rotation in the 1st round;
3. $2n$-bit outputs ($n = 3, 4, 5$) of $\mathrm{lsb}_n(A_{r+1})||\mathrm{lsb}_n(C_{r+1})$ lead to much stronger biases if $\mathrm{lsb}_5(A_0) = 0$, $\mathrm{lsb}_5(C_0) = 0$, and both B_0 and D_0 introduce zero rotation in the 1st round.

In other words, the previous key recovery algorithms make use of the distinguishing algorithms that fix $\mathrm{lsb}_n(A_0)$, $\mathrm{lsb}_n(C_0)$, or both and that introduce zero rotation in the 1st round. However, fixing the 1st-round rotation requires much memory for the key recovery attack and reduces the available number of plaintexts [7]. Here, in order to investigate other conditions that have almost the same effect but that do not reduce the available number of plaintexts, we conduct the following three experiments.

Test 1: The χ^2-test on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in the case of which $\mathrm{lsb}_5(A_0)||\mathrm{lsb}_5(C_0)$ is set to 0.
Test 2: The χ^2-test on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in the case of which $\mathrm{lsb}_5(B_0)||\mathrm{lsb}_5(D_0)$ is set to 0.
Test 3: The χ^2-test on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in the case of which $\mathrm{lsb}_4(B_0)||\mathrm{lsb}_4(D_0)$ is set to 0.

Test 1 corresponds to the previous χ^2-test [7,9]. Since we have known in [9] that the χ^2-value of $\mathrm{lsb}_n(A_{r+1})||\mathrm{lsb}_n(C_{r+1})$ ($n = 2, 3, 4$) outputs almost the same bias, we present only the results of $n = 3$ to compare the difference between $\mathrm{lsb}_5(A_0)||\mathrm{lsb}_5(C_0) = 0$ and $\mathrm{lsb}_5(B_0)||\mathrm{lsb}_5(D_0) = 0$. Test 2 or 3 fixes $\mathrm{lsb}_5(B_0)||\mathrm{lsb}_5(D_0)$ or $\mathrm{lsb}_4(B_0)||\mathrm{lsb}_4(D_0)$ instead of $\mathrm{lsb}_5(A_0)||\mathrm{lsb}_5(C_0)$, respectively. Our experiments generate all plaintexts by using M-sequence [8]. For example, 118-, 123-, and 128-bit random numbers are generated by M-sequence, whose primitive polynomials of M-sequence are $x^{118} + x^{36} + x^8 + x + 1$, $x^{123} + x^{16} + x^{12} + x + 1$, and $x^{128} + x^7 + x^2 + x + 1$, respectively. Our platforms are IBM RS/6000 SP (PPC 604e/332MHz $\times$ 256) with memory of 32 GB and PC cluster system (Pentium III/1GHz $\times$ 50) with memory of 12.5 GB. All tests use 10^3 keys and 10^2 kinds of plaintexts, and thus conduct 10^5 trials in total.

3.2 Test 1 and Test 2

The results of Tests 1 or 2 are shown in Tables 3 or 4, respectively. These results indicate that Test 1 outputs more bias than Test 2, but that Test 2 also outputs enough bias by using the same number of plaintexts. As reported in [9], we do not necessarily need much bias like level of 0.95 as in [7] to recover a correct key, which will be also shown in the subsequent sections. In fact, the level of more than 0.57 is enough for key recovering. Furthermore if we employ Test 1 to key recovery algorithm, the 1st-round rotation has to be fixed to zero in order to maintain the effect after post-whitening. However it requires extremely much memory. Considering these conditions, we employ Tests 2 and 3 to key recovery algorithm.

3.3 Test 2 and Test 3

Table 5 shows the results of Test 3. Tables 4 and 5 indicate that Test 2 outputs higher χ^2-value with fewer number of plaintexts than Test 3; but that Test 3 also outputs enough high bias.

Suppose that $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$ is fixed to some value except $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0) = 0$ ($n = 4, 5$). Then, $\mathrm{lsb}_n(A_2)||\mathrm{lsb}_n(C_2)$, i.e. $(\mathrm{lsb}_n(B_0) + \mathrm{lsb}_n(S_0)) \pmod{2^n}||(\mathrm{lsb}_n(D_0) + \mathrm{lsb}_n(S_1)) \pmod{2^n}$, is fixed in the same way as $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0) = 0$. Namely, whatever value $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$ ($n = 5, 4$) in Test 2 or 3 is fixed to, the same result as Table 4 or 5 is expected. Thus, we can generalize Test 2 or 3 to use any plaintext by just classifying it to each $\mathrm{lsb}_n(B_0)$ and $\mathrm{lsb}_n(D_0)$, and thus the number of available plaintexts in each Test is 2^{128}.

There is each naturally-extended key recovery attack that makes use of Test 2 or 3 as χ^2-test. In the next section, we apply Test 2 or 3 to the key reovery algorithm to RC6P, Algorithms 2 and 3, or 4. The number of available plaintexts of Algorithms 2 and 3, or 4 is 2^{118} and 2^{123}, or 2^{128}, respectively. These further differ in the number of *classifications*, which has an influence on the memory size or variance of key recovery attacks. *Classification* means the number of groups, in which plaintexts are classified and the average of χ^2-value is computed. In the subsequent sections, we will see how these differences work on each corresponding key recovery attack.

3.4 Slope

To extend our discussion on lower rounds to that on higher rounds, we estimate the slope of Tests 2 and 3 as in [7], that is, how many plaintexts are required to get similar values in a χ^2-test on $r + 2$ rounds compared with r rounds. Table 6 shows the number of plaintexts required for the χ^2-values with each level of 0.55, 0.60, 0.65, and 0.70, and estimates that each average slope of Test 2 (resp. Test 3) is $2^{16.01}$ (resp. $2^{16.03}$). Both Tests output almost the same slope, but Test 2 outputs slightly smaller slope than Test 3. This is because Test 2 fixes more bits of input than that of Test 3. In our estimation, we take each largest value $2^{16.04}$

or $2^{16.06}$ as each slope of Test 2 or 3 to make our estimation strict, respectively. In the following sections, we will show Algorithms 2 and 3 to RC6P, Algorithms 5 and 6 to RC6 (resp. Algorithm 4 to RC6P, Algorithm 7 to RC6), which are based on Test 2 (resp. Test 3). Each algorithm conducts the same χ^2-test as that of each corresponding Test. Therefore, to extend our discussion on lower rounds to that on higher rounds, we use the slope of each corresponding Test.

Table 7 shows the efficiency of each Test from the point of view of distinguishing attack. Considering the number of available plaintexts of Test 2 (resp. Test 3), 2^{118} (resp. 2^{120}), Test 2 (resp. Test 3) can distinguish output of 15-round RC6 from a randomly chosen permutation by using $2^{112.0}$ plaintexts (resp. $2^{112.90}$ plaintexts). Test 2 can work better than Test 3 from the point of view of distinguishing attack as we noted the above. In the subsequent sections, we will show some key recovery algorithms based on Test 2 or 3 that differ each other in the number of classifications.

Table 3. The χ^2-value on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in Test 1 (the average of 10^5 trials)

2 rounds				4 rounds			
# Texts	χ^2-value	Level	Variance	# Texts	χ^2-value	Level	Variance
2^8	63.402	0.538	126.731	2^{24}	63.489	0.541	127.840
2^9	63.875	0.554	129.299	2^{25}	64.028	0.560	129.847
2^{10}	64.729	0.584	133.864	2^{26}	65.006	0.593	134.789
2^{11}	66.415	0.640	142.293	2^{27}	67.052	0.660	144.714
2^{12}	69.939	0.744	157.668	2^{28}	71.000	0.771	167.825

Table 4. The χ^2-value on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in Test 2 (the average of 10^5 trials)

3 rounds				5 rounds			
# Texts	χ^2-value	Level	Variance	# Texts	χ^2-value	Level	Variance
2^8	63.224	0.532	125.883	2^{24}	63.262	0.533	126.990
2^9	63.416	0.538	126.119	2^{25}	63.429	0.539	127.497
2^{10}	63.819	0.553	129.069	2^{26}	63.790	0.552	128.212
2^{11}	64.669	0.582	132.916	2^{27}	64.521	0.578	131.408
2^{12}	66.352	0.638	140.551	2^{28}	66.373	0.639	140.554

4 Cryptanalysis against RC6 without Post-whitening

We present three key recovery algorithms against RC6P, and discuss their differences and the optimal condition to attack to RC6P. The main idea of these algorithms follow [9], but we fix some bits out of $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$ instead of

Table 5. The χ^2-value on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in Test 3 (the average of 10^5 trials)

3 rounds				5 rounds			
# Texts	χ^2-value	Level	Variance	# Texts	χ^2-value	Level	Variance
2^9	63.166	0.530	125.506	2^{25}	63.251	0.533	128.115
2^{10}	63.449	0.540	127.468	2^{26}	63.450	0.540	127.756
2^{11}	63.878	0.555	128.891	2^{27}	63.849	0.554	130.461
2^{12}	64.865	0.589	132.279	2^{28}	64.800	0.586	132.642
2^{13}	66.778	0.651	141.879	2^{29}	66.744	0.650	141.138

Table 6. $\log_2(\#\text{texts})$ and the slope required for the χ^2-value of each level (the average of 10^5 trials)

	Test 2			Test 3		
Level	3 rounds	5 rounds	slope	3 rounds	5 rounds	slope
0.55	9.92	25.89	15.97	10.80	26.83	16.03
0.60	11.45	27.49	16.04	12.26	28.32	16.06
0.65	12.17	28.17	16.00	13.00	29.00	16.00
0.70	12.71	28.72	16.01	13.53	29.57	16.04
average			16.01			16.03

Table 7. $\log_2(\#\text{texts})$ and linear equations for Tests 2 and 3 (the average of 10^5 trials)

	Test 2			Test 3		
Level	3 rounds	5 rounds	linear equation	3 rounds	5 rounds	linear equation
0.99	15.7	31.8	$8.02r - 8.30$	16.6	32.6	$8.03r - 7.55$

$\mathrm{lsb}_n(A_0)||\mathrm{lsb}_n(C_0)$ or the first-round-rotation amount. Intuitively, our algorithms fix some bits out of $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$, check the χ^2-value of $\mathrm{lsb}_3(A_r)||\mathrm{lsb}_3(C_r)$, and recover both $\mathrm{lsb}_2(S_{2r})$ and $\mathrm{lsb}_2(S_{2r+1})$ of r-round RC6P. Here we set $(y_b, y_d) = (\mathrm{lsb}_3(B_{r+1}), \mathrm{lsb}_3(D_{r+1}))$, $(x_c, x_a) = (\mathrm{lsb}_5(F(A_{r+1})), \mathrm{lsb}_5(F(C_{r+1})))$, $(s_a, s_c) = (\mathrm{lsb}_2(S_{2r}), \mathrm{lsb}_2(S_{2r+1}))$, $s = s_a||s_c$, and $(S^3_{2r}, S^3_{2r+1}) = (0, 0)$, where x_a (resp. x_c) is the rotation amounts on A_r (resp. C_r) in the r-th round.

4.1 Key Recovery Algorithms Based on Test 2

Algorithm 2 and 3 are based on Test 2 in Section 3. Algorithm 2 averages the χ^2-value among 2^{10} classifications, while Algorithm 3 averages it among 2^{15} classifications.

Algorithm 2

1. `Choose a plaintext` (A_0, B_0, C_0, D_0) `with` $(\mathtt{lsb}_5(B_0), \mathtt{lsb}_5(D_0)) = (0, 0)$ `and encrypt it.`
2. `For each` (s_a, s_c)`, decrypt` $y_d||y_b$ `with a key` $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ `by 1 round`[1].

[1] Since any (S^3_{2r}, S^3_{2r+1}) outputs the same χ^2-value of z [9], we may decrypt y by setting $(S^3_{2r}, S^3_{2r+1}) = (0, 0)$.

The decryptions of y_d and y_b are set to z_a and z_c, respectively, which are denoted by a 6-bit integer $z = z_a||z_c$.
3. For each value s, x_a, x_c, and z, update each array by incrementing $\mathrm{count}[s][x_a][x_c][z]$.
4. For each s, x_a, and x_c, compute $\chi^2[s][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][x_a][x_c]\}_{x_a,x_c}$ for each s and output s with the highest $ave[s]$ as $\mathrm{lsb}_2(S_{2r})||\mathrm{lsb}_2(S_{2r+1})$.

Algorithm 3

1. Choose a plaintext (A_0, B_0, C_0, D_0) with $\mathrm{lsb}_5(D_0) = 0$, set $t = \mathrm{lsb}_5(B_0)$, and encrypt it.
2. For each (s_a, s_c), decrypt $y_d||y_b$ with a key $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ by 1 round. The decryptions of y_d and y_b are set to z_a and z_c, respectively, which are also denoted by a 6-bit integer $z = z_a||z_c$.
3. For each value s, t, x_a, x_c, and z, update each array by incrementing $\mathrm{count}[s][t][x_a][x_c][z]$.
4. For each s, t, x_a, and x_c, compute $\chi^2[s][t][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][t][x_a][x_c]\}_{x_a,x_c,t}$ for each s and output s with the highest $ave[s]$ as $\mathrm{lsb}_2(S_{2r})||\mathrm{lsb}_2(S_{2r+1})$.

Table 8 shows the results of Algorithms 2 and 3 on 4-round RC6P: SUC, the average of χ^2-values $ave[s]$ on recovered keys, the level, and the variance, where SUC is the success probability among 1000 keys. Before comparing the results of Algorithms 2 and 3 (Table 8) with that of Test 2 (Table 4), we may review the fact of distribution of the mean [4], that is, for the mean μ or the variance σ^2 of a population, the mean or the variance of the distribution of the mean of a random sample with the size n drawn from the population are μ or σ^2/n, respectively. Plaintexts in Algorithm 2 or 3 are classified into 2^{10} or 2^{15} groups of $\{x_a, x_c\}$ or $\{\mathrm{lsb}_5(B_0), x_a, x_c\}$ and $ave[s]$ is computed over each group. On the other hand, all plaintexts are uniformly distributed to each group since they are randomly generated by M-sequences in our experiments. Therefore, the χ^2-value $ave[s]$ in Algorithm 2 or 3 is computed by using $1/2^{10}$ or $1/2^{15}$ times the number of plaintexts in Table 8. Applying this discussion to the experimental results, we see that the above fact of distribution of the mean exactly holds in Algorithms 2 and 3: the average of χ^2-value on $2^{18} - 2^{22}$ or $2^{23} - 2^{25}$ plaintexts in Algorithm 2 or 3 corresponds to that of $2^8 - 2^{12}$ or $2^8 - 2^{10}$ plaintexts in the case of $r = 3$ of Table 4; the variance of χ^2-values in Algorithm 2 or 3 corresponds to about $1/2^{10}$ or $1/2^{15}$ as much as that of Table 4; the averages of χ^2-values by using $2^{23} - 2^{25}$ plaintexts in Algorithm 3 are roughly equal to those by using $2^{18} - 2^{20}$ plaintexts in Algorithm 2; and the variances of χ^2-values by using $2^{23} - 2^{25}$ plaintexts in Algorithm 3 are about $1/2^5$ as much as those by using $2^{18} - 2^{20}$ plaintexts in Algorithm 2. We also remark that the level of χ^2-value more than 0.57 or 0.53 is enough for key recovering in Algorithm 2 or 3, respectively.

Let us discuss the security in higher rounds. Since Algorithms 2 and 3 are based on the χ^2-test of Test 2, we may expect that the slope in Test 2 holds in Algorithms 2 and 3. By using detailed experimental results in Table 9 and the

slope in Test 2, the number of plaintexts required for recovering a key in r-round RC6P with the success probability of 95%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = \begin{cases} 8.02r - 10.48 \text{ (Algorithm 2)} \\ 8.02r - 7.98. \text{ (Algorithm 3).} \end{cases}$$

Let us investigate the amount of work by setting one unit of work to one encryption. Algorithms 2 and 3 encrypts each plaintext and decrypts a ciphertext by 1 round with each key candidate. Therefore, the amount of work is $\#texts \times (1 + 1/r \times 2^4)$. Thus, by substituting the number of available plaintexts 2^{118} or 2^{123}, Algorithm 2 or 3 can break 16-round RC6P by using $2^{117.84}$ or $2^{120.34}$ plaintexts, $2^{118.84}$ or $2^{121.34}$ work, and 2^{20} or 2^{25} memory with a probability of 95%, respectively.

Table 8. The average of χ^2-value and the variance in Algorithms 2, 3, and 4 on 4-round RC6P (in 1000 trials)

Algorithm 2					Algorithm 3				
#texts	SUC	χ^2-value	Level	Variance	#texts	SUC	χ^2-value	Level	Variance
2^{18}	0.097	63.122	0.5280	0.1241	2^{21}	0.122	63.102	0.5273	0.0020
2^{19}	0.155	63.261	0.5329	0.1260	2^{22}	0.247	63.114	0.5278	0.0022
2^{20}	0.344	63.534	0.5425	0.1289	2^{23}	0.526	63.157	0.5293	0.0026
2^{21}	0.744	64.096	0.5621	0.1278	2^{24}	0.938	63.278	0.5336	0.0038
2^{22}	0.995	65.187	0.5994	0.1316	2^{25}	1.000	63.561	0.5435	0.0044

Algorithm 4				
#texts	SUC	χ^2-value	Level	Variance
2^{23}	0.117	63.011	0.5241	0.0003
2^{24}	0.177	63.020	0.5244	0.0004
2^{25}	0.347	63.037	0.5250	0.0004
2^{26}	0.768	63.067	0.5261	0.0005
2^{27}	1.000	63.139	0.5286	0.0005

Table 9. $\log_2(\#texts)$ required for key recovering of 4-round RC6P with each success probability (in 1000 trials)

Success	Algorithm 2			Algorithm 3			Algorithm 4		
Probability	#texts	χ^2-value	Level	#texts	χ^2-value	Level	#texts	χ^2-value	Level
95%	$2^{21.6}$	64.539	0.5778	$2^{24.1}$	63.295	0.5341	$2^{26.6}$	63.102	0.5273
50%	$2^{20.4}$	63.721	0.5507	$2^{23.0}$	63.157	0.5293	$2^{25.4}$	63.045	0.5253

4.2 Key Recovery Algorithm Based on Test 3

Algorithm 4 is based on the χ^2-test of Test 3 in Section 3 and averages it among 2^{18} classifications.

Algorithm 4

1. Choose a plaintext (A_0, B_0, C_0, D_0), set $(t_b, t_d) = (\mathtt{lsb}_4(B_0), \mathtt{lsb}_4(D_0))$, and encrypt it.
2. For each (s_a, s_c), decrypt $y_d||y_b$ with a key $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ by 1 round. The decryptions of y_d and y_b are set to z_a and z_c, which are also denoted by a 6-bit integer $z = z_a||z_c$.
3. For each value s, t_b, t_d, x_a, x_c, and z, update each array by incrementing $\mathtt{count}[s][t_b][t_d][x_a][x_c][z]$.
4. For each s, t_b, t_d, x_a, and x_c, compute $\chi^2[s][t_b][t_d][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][t_b][t_d][x_a][x_c]\}_{t_b,t_d,x_a,x_c}$ for each s, and output s with the highest $ave[s]$ as $\mathtt{lsb}_2(S_{2r})||\mathtt{lsb}_2(S_{2r+1})$.

Table 8 shows the results of Algorithm 4. Algorithm 4 classifies plaintexts into 2^{18} groups of $\{\mathrm{lsb}_4(B_0), \mathrm{lsb}_4(D_0), x_a, x_c\}$ and averages χ^2-value over each group. In the same discussion as Algorithms 2 and 3, we see that the average of χ^2-values by using 2^{27} plaintexts in Table 8 is roughly equal to that by using 2^9 plaintexts in the case of $r = 3$ of Table 5; and the variance of χ^2-values by using 2^{27} plaintexts in Table 8 is about $1/2^{18}$ as much as that by using 2^9 plaintexts in the case of $r = 3$ of Table 5. We note that the χ^2-value level of more than 0.527 is enough for key recovering in Algorithm 4.

Let us discuss the security in higher rounds. In the same discussion as Algorithms 2 and 3, we apply the slope of Test 3 in that of Algorithm 4. By using more detailed experimental results in Table 9 and the slope of Test 3, the number of plaintexts required for recovering a key in r-round RC6P with the success probability of 95%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = 8.03r - 5.52.$$

By substituting the number of available plaintexts 2^{128}, Algorithm 4 can break 16-round RC6P by using $2^{122.96}$ plaintexts, $2^{123.96}$ work, and 2^{28} memory with a probability of 95%.

4.3 Comparison of Algorithms 2, 3, and 4

Algorithms 2, 3, and 4 differ mainly in the number of classifications. In other words, they differ in the number of plaintexts that the χ^2-values are averaged. We investigate how such a difference influences on a key recovery algorithm. Table 10 summarizes results of three algorithms: the applicable rounds and the efficiency. Algorithm 4 can break 16-round RC6P in the success probability of 95% with the lowest level of χ^2-value, at most 0.528. Because, the more the number of classifications is, the smaller the variance of χ^2-value are, as we reviewed the fact above. The smaller variance is one of necessary factors to single out a correct key as in [9]. However, in contrast to [9], Algorithm 4 is not the most efficient attack of three algorithms. Three algorithms can analyze RC6P with the same number of rounds. That is, it does not necessarily holds that the more number of classifications, the larger applicable rounds. Generally, the larger the number of classifications, the lower level of χ^2-value are required to recover a correct key but the more necessary plaintexts and work are required. On the other hand,

there exists an upper limit of the available plaintexts and work amount. This is why the optimization of the number of classifications is necessary.

There are two factors of the number of both available texts and classifications to discuss the optimization. Fixing the number of available texts to 2^{128}, let us investigate the optimal number of classifications: the χ^2-value is averaged over groups $\{\mathrm{lsb}_3(B_0), \mathrm{lsb}_3(D_0), x_a, x_c\}$, $\{\mathrm{lsb}_4(B_0), \mathrm{lsb}_4(D_0), x_a, x_c\}$, or $\{\mathrm{lsb}_5(B_0), \mathrm{lsb}_5(D_0), x_a, x_c\}$, namely the number of classifications is 2^{16}, 2^{18}, or 2^{20}, respectively. This means that we optimize Algorithm 4 by changing the number of classification. Table 11 shows the results, which indicates that the key recovery attack with 2^{18} classifications, i.e. Algorithm 4, is the optimal. The number of classifications of Algorithms 2 and 3 is also optimized to attack RC6 well.

Table 10. Comparison of Algorithms 2, 3, and 4 on RC6P: applicable rounds and the efficiency

Algorithm	#classifications	#available texts	nemory	rounds	#texts	work	χ^2-value (level)	variance
2	2^{10}	2^{118}	2^{20}	16	$2^{117.84}$	$2^{118.84}$	64.539 (0.578)	0.1319
3	2^{15}	2^{123}	2^{25}	16	$2^{120.34}$	$2^{121.34}$	63.295 (0.535)	0.0039
4	2^{18}	2^{128}	2^{28}	16	$2^{122.96}$	$2^{123.96}$	63.102 (0.528)	0.0005

Table 11. #texts necessary for variations of Algorithm 4 on 4-round RC6P

	#classifications		
	2^{16}	2^{18}	2^{20}
#texts (95%)	$2^{27.0}$	$2^{26.6}$	$2^{26.9}$
#texts (50%)	$2^{25.8}$	$2^{25.4}$	$2^{25.7}$

5 Cryptanalysis against RC6

In this section, we apply Algorithm 2, 3, or 4 to RC6 with a 24-byte key, which is called Algorithm 5, 6, or 7, respectively. They recover a 68-bit subkey of $\mathrm{lsb}_2(S_{2r})$, $\mathrm{lsb}_2(S_{2r+1})$, S_{2r+2}, and S_{2r+3}. We demonstrate Algorithm 5 to RC6-8 and discuss how to analyze the security to RC6 with a 24-byte key.

5.1 Attacks on RC6

Let us set $(y_b, y_d) = (\mathrm{lsb}_3(B_{r+2}), \mathrm{lsb}_3(D_{r+2}))$, $(s_a, s_c) = (\mathrm{lsb}_2(S_{2r}), \mathrm{lsb}_2(S_{2r+1}))$, $s = s_a||s_c||S_{2r+2}||S_{2r+3}$, $(S^3_{2r}, S^3_{2r+1}) = (0, 0)$, and $(x_c, x_a) = (\mathrm{lsb}_5(F(A_{r+2} - S_{2r+2})), \mathrm{lsb}_5(F(C_{r+2} - S_{2r+3})))$, where x_a or x_c is the r-round rotation amounts on A_r or C_r, respectively.

Algorithm 5

1. Choose a plaintext (A_0, B_0, C_0, D_0) with $(\mathtt{lsb}_5(B_0), \mathtt{lsb}_5(D_0)) = (0, 0)$ and encrypt it.
2. For each subkey S_{2r+2} and S_{2r+3}, decrypt $y_d||y_b$ with a key $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ by 1 round. The decryptions of y_d and y_b are set to z_a and z_c, respectively, which are denoted as a 6-bit integer $z = z_a||z_c$.
3. For each value s, x_a, x_c, and z, update each array by incrementing $\mathtt{count}[s][x_a][x_c][z]$.
4. For each s, x_a, and x_c, compute $\chi^2[s][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][x_a][x_c]\}_{x_a,x_c}$ for each s, and output s with the highest $ave[s]$ as $\mathtt{lsb}_2(S_{2r})||\mathtt{lsb}_2(S_{2r+1})||S_{2r+2}||S_{2r+3}$.

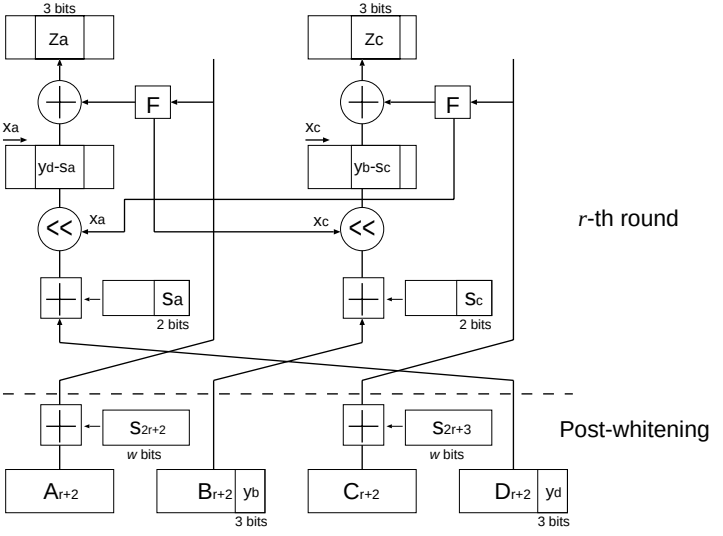

Fig. 1. Outline of Algorithm 5

Figure 1 shows the outline of Algorithm 5. Algorithm 5 differ with Algorithm 2 in a way of handling both S_{2r+2} and S_{2r+3}: Algorithm 2 uses a correct key on S_{2r+2} and S_{2r+3}; but Algorithm 5 has to guess a correct key of S_{2r+2} and S_{2r+3}. Therefore, the results of Algorithm 5 against r-round RC6 is coincident with those of Algorithm 2 against r-round RC6P whenever correct keys on S_{2r+2} and S_{2r+3} are used. As a result, to discuss the security on RC6 against Algorithm 5, we have only to investigate the behavior of χ^2-value with using wrong-keys of S_{2r+2} and S_{2r+3}.

5.2 Differences between Algorithms 2 and 5

To investigate the difference between two algorithms, let us observe how wrong-keys of S_{2r+2} have an influence on a key recovery in Algorithm 5 when a correct key is set to S_{2r+3}. Table 12 shows the experimental results of Algorithm 2 on RC6P-8 or Algorithm 5 on RC6-8, in which Algorithm 2 recovers 4-bit subkeys of $\mathrm{lsb}_2(S_8)$ and $\mathrm{lsb}_2(S_9)$; and Algorithm 5 recovers 12-bit subkeys of $\mathrm{lsb}_2(S_8)$, $\mathrm{lsb}_2(S_9)$, and S_{10}. Table 12 indicates that: Algorithm 5 cannot work as effectively as Algorithm 2 if a few plaintexts like 2^{11} or 2^{12} are used; but Algorithm 5 can work as effectively as Algorithm 2 if enough many plaintexts like 2^{14} or 2^{15} are used. They differ in the number of wrong keys: the number of wrong keys of Algorithm 5 is 2^8 times as many as that of Algorithm 2. If a few (i.e. not enough) plaintexts are used, then the χ^2-value on even a correct key is rather low and thus the χ^2-value on wrong keys disturbs us to single out a correct key. As a result, the difference in the number of wrong keys influences the probability that can single out a correct key. On the other hand, if *enough* number of plaintexts are used, then the χ^2-value on a correct key becomes enough high, while that on wrong keys does not become high, and, thus, the difference in the number of wrong keys does not have a great influence on singling out a correct key. As a result, Algorithm 5 can single out a correct key with almost the same high probability like more than 90% as Algorithm 2 if *enough* number of plaintexts are used. The remaining problem is how to define *enough* number of plaintexts. We may note that the key recovery attacks compute the χ^2-value on a part for every key candidate and output a key with the highest χ^2-value as a correct key. This means that an algorithm can single out a correct key if and only if a correct key outputs higher χ^2-value than that on all wrong keys. In other words, the lowest χ^2-value on correct keys has only to be higher than the highest χ^2-value on wrong keys. Thus, *enough* number of plaintexts necessary to single out a correct key is defined as the number of plaintexts that makes the lowest χ^2-value on correct keys higher than the highest χ^2-value on wrong keys.

As the final step, we investigate a good sample on wrong keys of S_a, S_c, S_{2r+2} and S_{2r+3} that may output the the highest χ^2-value. Let us set the *almost-correct wrong key* that differs a correct key in only the most-significant-one bit of S_{2r+2}: the other bits, in other words, S_a, S_c, S_{2r+3} and $\mathrm{lsb}_7(S_{2r+2})$, are the same as a correct key. Apparently, this is the most similar to a correct key and is expected to output the highest χ^2-value of wrong keys. Thus, we define enough number of plaintexts to single out a correct key as the number of plaintexts such that the lowest χ^2-value on correct keys becomes higher than the highest χ^2-value on almost-correct wrong keys. To find out enough number of plaintexts in the case of Algorithm 5 on RC6-8, we conduct the following two experiments:

- **Test 4**[2]**: [Behavior of** $\chi^2 - value$ **of correct keys]**
  Compute the highest and lowest χ^2-value on correct keys.
- **Test 5: [Behavior of** $\chi^2 - value$ **of almost-correct wrong keys]**
  Compute the highest χ^2-value on almost-correct wrong keys.

[2] Test 4 is the same as the results of correct keys in Algorithm 2 to RC6P.

The results are shown in Table 13, where SUC means the success probability to recover a correct key of S_a and S_b in Algorithm 2 to RC6P-8. From Table 13, we see that enough number of plaintexts is defined as $2^{14.5}$ plaintexts. Comparing with Table 12, we convince that enough number is well difined and, thus, we estimate that Algorithm 5 can recover a correct key with the success probability of about 90% by using $2^{14.5}$ plaintexts. Table 13 also indicates that the χ^2-value recovered by almost-correct wrong keys does not become high even if many plaintexts are used. This reflects that: the f-function of RC6 is the nonlinear conversion which depends on all 32-bit inputs; and thus the recovered value does not output high χ^2-value if only the input of f-function differs with a correct input even in 1 bit.

Table 12. Success probability of Algorithm 2 (resp. 5) on 4-round RC6P-8 (resp. RC6-8) (in 1000 trials)

#texts	Algorithm 2	Algorithm 5
2^{11}	0.125	0.001
2^{12}	0.241	0.014
2^{13}	0.486	0.177
2^{14}	0.887	0.886
2^{15}	1.000	1.000

Table 13. Results of Tests 4 and 5 in Algorithm 5 on 4-round RC6-8 (in 1000 trials)

		Test 4		Test 5
#texts	SUC	the highest χ^2	the lowest χ^2	the highest χ^2
$2^{12.0}$	0.232	69.711	60.689	68.634
$2^{12.5}$	0.332	70.435	61.680	67.794
$2^{13.0}$	0.491	72.167	62.226	68.181
$2^{13.5}$	0.699	74.458	63.266	67.450
$2^{14.0}$	0.868	77.427	65.359	68.393
$2^{14.5}$	0.972	81.609	68.971	68.174
$2^{15.0}$	0.999	88.978	70.890	68.211

5.3 The Security of RC6 against Algorithms 5

The previous section have defined enough number of plaintexts and seen that Algorithm 5 can recover a correct key with the success probability of 90% by using enough many plaintexts. We conduct Tests 4 and 5 to Algorithm 5 on 4-round RC6 to find out enough number of plaintexts. The results are shown in Table 14, where SUC means the success probability to recover a correct key of S_a and S_b in Algorithm 2 to RC6P. Table 14 indicates that enough number of plaintexts is set to 2^{22} plaintexts; and it is roughly equal to that which outputs

the success probability of more than 95% in Algorithm 2 on RC6P. Then, we estimate that Algorithm 5 can recover a correct key with the success probability of more than 90% by using 2^{22} plaintexts.

Let us discuss the security in higher-round RC6. We increase the number of plaintexts by up to a factor of 2^2 to analyze the security strictly and use the same slope in Test 2 since Algorithm 5 is based on the χ^2-test of Test 2. Then, the number of plaintexts required for recovering a key in r-round RC6 with the success probability of 90%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = 8.02r - 8.08.$$

By substituting the number of available plaintexts 2^{118}, Algorithm 5 can break 15-round RC6 by using $2^{112.22}$ plaintexts with a probability of about 90%.

5.4 Applying Algorithms 3 to RC6

Algorithm 3 can be applied to RC6 in the same way as Algorithm 2, which is called Algorithms 6. We omit the repetitious detail of the algorithm. To find out enough number of plaintexts, we conduct the same experiments of Tests 4 and 5 to Algorithm 6 on 4-round RC6, whose results are shown in Table 14. Table 14 indicates that enough number of plaintexts to Algorithm 6 on RC6 is $2^{24.6}$ plaintexts.

We increase the number of plaintexts by up to a factor of 2^2 to analyze the security strictly and use the slope of Test 2 in Section 3. Then, the number of plaintexts required for a key recovering in r-round RC6 with the success probability of 90%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = 8.02r - 5.48.$$

By substituting the number of available plaintexts 2^{123}, Algorithms 6 can break 16-round RC6 with 192-bit key by using $2^{122.84}$ plaintexts with a probability of 90%.

Let us compare Algorithms 5 and 6 from the point of view of the number of plaintexts and the amount of work, where one unit of work is set to one encryption. Both algorithms encrypt each plaintext, and decrypt a ciphertext by 1 round with each key candidate, where the number of key candidates is 2^{68}. Thus, the amount of work is computed by $\#texts \times (1 + 2^{68}/r)$. These results are shown in Table 15. In the final paper, the optimization of each algorithm including the results of Algorithm 7 will be discussed.

6 Conclusion

In this paper, we have discussed the optimization of the number of classification by presenting three key recovery attacks against RC6P, Algorithms 2, 3, and 4. As a result of optimization, Algorithm 2 can break 16-round RC6P by

Table 14. Results of Tests 4 and 5 in Algorithms 5 and 6 on 4-round RC6 (in 1000 trials)

Algorithm 5					Algorithm 6				
		Test 4					Test 5		
#texts	SUC	the highest	the lowest	Test 5	#texts	SUC	the highest	the lowest	Test 5
$2^{20.0}$	0.344	63.534	62.644	62.994	$2^{22.5}$	0.385	63.288	62.916	63.197
$2^{20.5}$	0.539	63.769	62.743	62.979	$2^{23.0}$	0.519	63.338	62.905	63.211
$2^{21.0}$	0.744	64.096	62.865	62.976	$2^{23.5}$	0.752	63.395	62.963	63.199
$2^{21.5}$	0.946	64.540	62.904	62.991	$2^{24.0}$	0.934	63.501	63.013	63.218
$2^{22.0}$	0.995	65.187	63.038	62.978	$2^{24.6}$	1.000	63.648	63.223	63.201

Table 15. #Texts and Work for Algorithms 5 and 6 on r-round RC6 (Estimated)

Algorithm	initial Key	rounds	#texts	work	linear equation (SUC$\geqq$90%)
5	192-bit	15	$2^{112.22}$	$2^{176.32}$	$8.02r - 8.08$
6	192-bit	16	$2^{122.84}$	$2^{186.84}$	$8.02r - 5.48$

using $2^{117.84}$ plaintexts with the success probability of 95%. We have also investigated how to estimate the security of RC6 to these key recovery algorithms by introducing the idea of *enough* number of plaintexts and *almost-correct* wrong key. We have shown that Algorithm 6 is estimated to break 16-round RC6 with 24-byte keys by using $2^{122.84}$ plaintexts with the success probability of 90%.

References

1. J. Borst, B. Preneel, and J. Vandewalle, "Linear Cryptanalysis of RC5 and RC6," Proc. Fast Software Encryption, LNCS 1636, pp.16–30, 1999.
2. S. Contini, R. Rivest, M. Robshaw, and Y. Yin, "The Security of the RC6 Block Cipher. v 1.0," August 20, 1998. Available at http://www.rsasecurity.com/rsalabs/rc6/.
3. S. Contini, R. Rivest, M. Robshaw, and Y. Yin, "Improved analysis of some simplified variants of RC6," Proc. Fast Software Encryption, LNCS 1636, pp.1–15, 1999.
4. R.J. Freund and W.J, Wilson, *Statistical Method*, Academic Press, San Diego, 1993.
5. H. Gilbert, H. Handschuh, A. Joux, and S. Vaudenay, "A Statistical Attack on RC6," Proc. Fast Software Encryption, LNCS 1978, pp.64–74, 2000.
6. H. Handschuh and H. Gilbert, "χ^2 Cryptanalysis of the SEAL Encryption Algorithm," Proc. Fast Software Encryption, LNCS 1267, pp.1–12, 1997.
7. L. Knudsen and W. Meier, "Correlations in RC6 with a reduced number of rounds," Proc. Fast Software Encryption, LNCS 1978, pp.94–108, 2001.
8. A. Menezes, P.C. van Oorschot, and S. Vanstone, Handbook of applied cryptography, CRC Press, Inc., Boca Raton, 1996.
9. A. Miyaji and M. Nonaka, "Cryptanalysis of the Reduced-Round RC6," Proc. ICICS 2002, LNCS 2513 pp.480–494, 2002.
10. R. Rivest, "The RC5 Encryption Algorithm," Proc. Fast Software Encryption, LNCS 1008, pp.86–96, 1995.

11. R. Rivest, M. Robshaw, R. Sidney, and Y. Yin, "The RC6 Block Cipher. v1.1," August 20, 1998. Available at http://www.rsasecurity.com/rsalabs/rc6/.
12. T. Shimoyama, M. Takenaka, and T. Koshiba, "Multiple linear cryptanalysis of a reduced round RC6," Proc. Fast Software Encryption, LNCS 2365, pp.76–88. 2002.
13. T. Shimoyama, K. Takeuchi, and J. Hayakawa, "Correlation Attack to the Block Cipher RC5 and the Simplified Variants of RC6," 3rd AES Candidate Conference, April 2000.
14. S. Vaudenay, "An Experiment on DES Statistical Cryptanalysis," Proc. 3rd ACM Conference on Computer and Communications Security, ACM Press, pp.139–147, 1996.

Anonymity-Enhanced Pseudonym System

Yuko Tamura and Atsuko Miyaji

1-1, Asahidai, Tatsunokuchi, Ishikawa, 923-1292, Japan
{yuko, miyaji}@jaist.ac.jp

Abstract. Pseudonym systems allow users to interact with multiple organizations anonymously by using pseudonyms. Such schemes are of significant practical relevance because it is the best means of providing privacy for users. In previous works, users transact with a organization by demonstration of possession of a credential issued by the organization or relationship with another credential. However, the information that a user has a credential from a specific organization compromises privacy of the user. In the present paper, we give a formal definition of practical pseudonym system in which the level of privacy provided can be chosen be according to security policies.

1 Introduction

As information gets increasingly accessible, it has been important that individuals control their information to protect their privacy. Pseudonym systems (also called anonymous credential systems) [1,2,3,4,5,6] allow users to work effectively and anonymously with multiple organizations by using different *pseudonyms*. Such systems are called anonymous when transactions carried out by the same user cannot be correlated. In the systems, an organization knows users by only pseudonym in which each pseudonym cannot be linked to others. An organization issues a *credential* on a pseudonym, and the corresponding user demonstrates the possession of this credential to another organization without revealing anything but the possession. Lysyanskaya, Rivest, Sahai and Wolf [5] proposed a general pseudonym system based on one-way functions and general zero-knowledge proofs. In their scheme, however, credentials for a user need to be reissued by the organization so that the user can prove the possession of a credential several times. Camenisch and Lysyanskaya [6] solved such a problem by applying strong-RSA-based signature schemes [14] and group signature schemes [9] to their pseudonym system. In their scheme, users can demonstrate the possession of credentials in any number of times and these demonstrations cannot be linked to the same pseudonym.

However, unfortunately, the previous schemes [1,2,3,4,5,6], proving the possession of a credential cannot but give a verifier the information about which organization a user transacts with. A pseudonym system by Camenisch and Lysyanskaya [6] is the most efficient and the practical in previous works. In their system, a user establishes a pseudonym and its validating tag, then the user is issued a credential as a signature on the tag by the organization. Their system

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 33–47, 2003.

requires the public-key of an issuing organization to prove the possession of a credential and thus gives the information of the organization to a verifier necessarily. As a result, this compromises the privacy of users, although the verifier does not necessarily need the information. The best pseudonym system should be able to choose the level of privacy according to its security policies.

In this paper, we propose an anonymity-enhanced pseudonym system by showing a credential issued by a group. Our system can allow a user to choose the level of privacy according to their security policies. In our system, an organization is a member of a group, a credential on a pseudonym with an organization is issued by the group manager. Such a credential is a signature on the validating tag and the public-key of the organization by the group manager. Consequently, the user can prove the possession of a credential from some organization in the group without informing of the organization. Moreover, our system provides a feature: *flexibility of choosing the methods to prove the possession of a credential* according to security policies. Namely, a user is given the four methods to prove: (1) showing a credential *with* the identity of an organization if a user needs to inform a verifier of the possession of a credential from the organization, (2) showing a credential *without* the identity of an organization if a user wants to give a verifier only the information that the credential is issued from a group, (3) transferring a credential *with* the identity of an organization and (4) transferring a credential *without* the identity of an organization if a user want to prove the possession of a credential to another organization with whom the user has established a pseudonym. Furthermore, our system satisfies all the desirable properties that the previous schemes [1,2,3,4,5,6] have.

The rest of this paper is organized as follows. The next section presents the formal definitions and the requirements of an anonymity-enhanced pseudonym system. In section 3, we propose a practical pseudonym system, after an overview. The security is discussed in section 4.

2 Formal Definitions and Requirements

2.1 The Model of Pseudonym System

Our pseudonym system is constituted by the following players:

Certification authority (CA) : only entity that knows the user's identity.
Group $(\boldsymbol{G}_I)$: set of organizations.
Group manager (G_I) : only entity that has a secret-key of $\boldsymbol{G}_I$ and grants a credential to a user. (We use CA and $G_0 \in \boldsymbol{G}_0$ as interchangeable name.)
Organization (O_i) : entity which belongs to groups.
User (U) : entity who registers with a group and transacts with an organization in the group by the pseudonym.
Verifier (V) : entity that verifies credentials of users.

Pseudonym systems should satisfy the following properties:

Anonymity of users : Verifiers (Verifying organizations) cannot find out anything about a user, except the fact of the user's ownership of a credentials, even if it cooperates with others.

Unlinkability of pseudonyms : Different pseudonyms of the same user are not linked, even if a group manager or an organization cooperates with others.

Unforgeability of credentials : It is impossible to forge a credential issued by a group manager, even if users, other group managers and organizations team up.

2.2 Ideal Credential System

We define an ideal pseudonyms system [6] that relies on a trusted party T as an intermediator that is responsible for the necessary properties of the system. All transactions are made via T. T also ensures anonymity of the users towards the group managers, organizations, and verifiers. For an ideal pseudonym system (IPS) and a cryptographic pseudonym system without T (CPS), we gives a security definition, same as [6].

Definition 1 *Let $V = poly(k)$ be the number of players in the system with security parameter k. For an ideal pseudonym system IPS, and its cryptographic implementation CPS, we denote a credential system with security parameter k and event scheduler E for the events that take place in this system, by $IPS(1^k, E)$ (resp., $(CPS(1^k, E))$). If $\{A_1(1^k), \ldots, A_V(1^k)\}$ is a list of the players's outputs, then we denote these player's outputs by $\{A_1(1^k), \ldots, A_V(1^k)\}^{PS(1^k,E)}$ when all of them, together, exist within a pseudonyms system PS. CPS is secure if there exists a simulator $\mathcal{S}$ (ideal-world adversary) such that the following holds, for all interactive probabilistic polynomial-time machines $\mathcal{A}$ (real-world adversary), for all sufficiently large k:*
(1) In the IPS, $\mathcal{S}$ controls the players in the ideal-world corresponding to those real world players controlled by $\mathcal{A}$.
(2) For all event schedulers $E^{\mathcal{A}}$

$$\{\{Z_i(1^k)\}_{i=1}^{V}, \mathcal{A}(1^k)\}^{CPS(1^k,E)} \stackrel{c}{\approx} \{\{Z_i(1^k)\}_{i=1}^{V}, \mathcal{S}^{\mathcal{A}}(1^k)\}^{IPS(1^k,E)}$$

where $\mathcal{S}$ is given black-box access to $\mathcal{A}$, ($D_1(1^k) \stackrel{c}{\approx} D_2(1^k)$ denotes computational indistinguishability of the distributions D_1 and D_2.)

2.3 Functional Definitions

This section provides functional definitions in our pseudonym system. Let k be the security parameter and $neg(k)$ denote any function that vanishes faster than any inverse polynomial.

Definition 2 *A pseudonym system consists of the following procedure:*

Key generation– GK_G, $GK_{(O,G)}$ *and* GK_U *for* $G, O \in \boldsymbol{G}$ *and* U *output a secret and public-key pair* $(\mathcal{X}_G, \mathcal{Y}_G)$ *for a group* $\boldsymbol{G}$, $(\mathcal{X}_{(O,G)}, \mathcal{Y}_{(O,G)})$ *for an organization* $O \in \boldsymbol{G}$, *and* $(\mathcal{X}_U, \mathcal{Y}_U)$ *for a user* U, *respectively.* GK_G *and* GK_U *take as input* 1^k, *and* $GK_{(U,O)}$ *takes* 1^k *and a group public-key* $\mathcal{Y}_G$.

Pseudonym generation– $GP\langle U, X\rangle$ *between* U *and an entity* $X \in \boldsymbol{G}$, *takes as* U*'s private input the secret-key* $\mathcal{X}_U$, *and as their common input a group public-key* $\mathcal{Y}_G$. *The private output for* U *is some secret information* $\mathcal{S}_{(U,X)}$, *and the common output is* U*'s pseudonym* $\mathcal{P}_{(U,X)}$.

Credential issue– $IC\langle U, G\rangle$ *between* U *and* $G \in \boldsymbol{G}$, *outputs a credential* $\mathcal{C}_{(U,G)}$ *on* $\mathcal{P}_{(U,G)} \in GP\langle U, G\rangle$. U*'s private input is* $\mathcal{X}_U$ *and* $\mathcal{S}_{(U,G)}$, G*'s private input is a group secret-key* $\mathcal{X}_G$, *and their common input is* $\mathcal{Y}_G$ *and* $\mathcal{P}_{(U,G)}$. $(GP \ni \mathcal{P}$ *means that* GP *outputs* $\mathcal{P}.)$

Pseudonym's validity generation– $GV\langle U, O_i\rangle$ *between* U *and* $O_i \in \boldsymbol{G}$, *outputs a signature on* $\mathcal{P}_{(U,O_i)} \in GP\langle U, O_i\rangle$. O_i*'s private input is a secret-key* $\mathcal{X}_{(O_i,G)}$, *and their common input is* $\mathcal{Y}_G, \mathcal{Y}_{(O_i,G)}$ *and* $\mathcal{P}_{(U,O_i)}$ *with* O_i. U*'s private output is a signature* $\sigma_{(U,O_i)}$.

Credential blind issue– $BIC\langle U, G\rangle$, *blind issue of a credential on a pseudonym, between* U *and* $G \in \boldsymbol{G}$, *outputs a credential* $\mathcal{C}_{(U,O_i)}$ *on* $\mathcal{P}_{(U,O_i)} \in GP\langle U, O_i\rangle$ *where* $O_i \in \boldsymbol{G}$. U*'s private input is* $\mathcal{X}_U, \mathcal{S}_{(U,G)}, \mathcal{S}_{(U,O_i)}$ *and* $\mathcal{P}_{(U,O_i)}$, G*'s private input is* $\mathcal{X}_G$, *and their common input is* $\mathcal{Y}_G, \mathcal{Y}_{(O_i,G)}, \mathcal{P}_{(U,G)}$ *and* $\sigma_{(U,O_i)}$.

Credential showing– $SC\langle U, V\rangle$, *showing a credential on a pseudonym with a group, between* U *and* V, *takes as* U*'s private input* $\mathcal{X}_U, \mathcal{S}_{(U,G)}, \mathcal{P}_{(U,G)}$ *and* $\mathcal{C}_{(U,G)}$, *and as their common input* $\mathcal{Y}_G$. *It outputs* 1 *or* 0, *which, if* $\mathcal{C}_{(U,G)} \in IC\langle U, G\rangle(\mathcal{P}_{(U,G)})$ *where* $\mathcal{P}_{(U,G)} \in GP\langle U, G\rangle$ *or not with probability* $1 - neg(k)$, *respectively.* $(IC(\mathcal{P}) \ni \mathcal{C}$ *means that* IC *outputs* $\mathcal{C}$ *by an input* $\mathcal{P}.)$ $SC^+\langle U, V\rangle$, *showing a credential with identity of an organization, between* U *and* V, *takes as* U*'s private input* $\mathcal{X}_U, \mathcal{S}_{(U,O_i)}, \mathcal{P}_{(U,O_i)}$ *and* $\mathcal{C}_{(U,O_i)}$, *and as their common input* $\mathcal{Y}_G$ *and* $\mathcal{Y}_{(O_i,G)}$. *It outputs* 1 *or* 0, *which, if* $\mathcal{C}_{(U,O_i)} \in BIC\langle U(\mathcal{P}_{(U,O_i)}), G\rangle$ *where* $\mathcal{P}_{(U,O_i)} \in GP\langle U, O_i\rangle$, *or not with probability* $1-neg(k)$, *respectively.* $(BIC\langle U(\mathcal{P}), G\rangle) \ni \mathcal{C}$ *means that* BIC *outputs* $\mathcal{C}$ *by* U*'s private input* $\mathcal{P}.)$

$SC^-\langle U, V\rangle$, *showing a credential without identity of an organization, between* U *and* V, *takes as* U*'s private input* $\mathcal{X}_U, \mathcal{S}_{(U,O_i)}, \mathcal{P}_{(U,O_i)}, \mathcal{C}_{(U,O_i)}$ *and* $\mathcal{Y}_{(O_i,G)}$, *and as their common input* $\mathcal{Y}_G$. *It outputs* 1 *or* 0, *which, if* $\mathcal{C}_{(U,O_i)} \in BIC\langle U(\mathcal{P}_{(U,O_i)}), G\rangle$ *where* $\mathcal{P}_{(U,O_i)} \in GP\langle U, O_i\rangle$, *or not with probability* $1 - neg(k)$, *respectively.*

Credential transfer– $TC\langle U, X_j\rangle$, *transferring a credential on a pseudonym with a group, between a user* U *and an entity* $X_j \in \boldsymbol{G}_J$, *takes as* U*'s private input* $\mathcal{X}_U, \mathcal{S}_{(U,G_I)}, \mathcal{S}_{(U,X_j)}, \mathcal{P}_{(U,G_I)}$ *and* $\mathcal{C}_{(U,G_I)}$, *as their common input* $\mathcal{Y}_{G_I}, \mathcal{Y}_{G_J}$ *and* $\mathcal{P}_{(U,X_j)}$. *It outputs* 1 *or* 0, *which, if* $\mathcal{C}_{(U,G_I)} \in IC\langle U, G_I\rangle(\mathcal{P}_{(U,G_I)})$, $\mathcal{P}_{(U,G_I)} \in GP\langle U(\mathcal{X}_U), G_I\rangle$ *and* $\mathcal{P}_{(U,X_j)} \in GP\langle U(\mathcal{X}_U), X_j\rangle$ *or not with probability* $1 - neg(k)$, *respectively.*

$TC^+\langle U, X_j\rangle$, *transferring a credential with identity of an organization, between* U *and* $X_j \in \boldsymbol{G}_J$, *takes as* U*'s private input* $\mathcal{X}_U, \mathcal{S}_{(U,O_i)}$, $\mathcal{S}_{(U,X_j)}, \mathcal{P}_{(U,O_i)}$ *and* $\mathcal{C}_{(U,O_i)}$, *as their common input* $\mathcal{Y}_{G_I}, \mathcal{Y}_{G_J}, \mathcal{Y}_{(O_i,G_I)}$ *and* $\mathcal{P}_{(U,X_j)}$. *It outputs* 1 *or* 0, *which, if* $\mathcal{C}_{(U,O_i)} \in BIC\langle U(\mathcal{P}_{(U,O_i)}), G_I\rangle$, $\mathcal{P}_{(U,O_i)} \in$

$GP\langle U(\mathcal{X}_U), O_i\rangle$, and $\mathcal{P}_{(U,X_j)} \in GP\langle U(\mathcal{X}_U), X_j\rangle$ or not with probability $1 - neg(k)$, respectively.
$TC^-\langle U, X_j\rangle$, transferring a credential without identity of an organization, between U and an entity $X_j \in \boldsymbol{G}_J$, takes as U's private input $\mathcal{X}_U, \mathcal{S}_{(U,O_i)}$, $\mathcal{S}_{(U,X_j)}, \mathcal{P}_{(U,O_i)}, \mathcal{C}_{(U,O_i)}$ and $\mathcal{Y}_{(O_i,G_I)}$, and as their common input $\mathcal{Y}_{G_I}, \mathcal{Y}_{G_J}$ and $\mathcal{P}_{(U,X_j)}$. It outputs 1 or 0, which, if $\mathcal{C}_{(U,O_i)} \in BIC\langle U(\mathcal{P}_{(U,O_i)}), G_I\rangle$, $\mathcal{P}_{(U,O_i)} \in GP\langle U(\mathcal{X}_U), O_i\rangle$, and $\mathcal{P}_{(U,X_j)} \in GP\langle U(\mathcal{X}_U), X_j\rangle$ or not with probability $1 - neg(k)$, respectively.

2.4 Notations

We use the same notation in [6,9] for the various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. (I) *Proof of knowledge or equality in different groups*: We use proofs that the discrete logarithms of two group elements $y_1 \in G_1, y_2 \in G_2$ to the bases $g_1 \in G_1$ and $g_2 \in G_2$ in different groups G_1 and G_2 which has an order q_1 and q_2, respectively, are equal . This proof can be realized only if both discrete logarithms lie in the interval $[0, min\{q_1, q_2\}]$. $PK\{(\alpha) : y_1 = {g_1}^\alpha \wedge y_2 = {g_2}^\alpha \wedge \alpha \in [0, min\{q_1, q_2\}]\}$ denotes a "*zero-knowledge proof of knowledge of integers α such that $y_1 = {g_1}^\alpha$ and $y_2 = {g_2}^\alpha$ holds, where $\alpha \in [0, min\{q_1, q_2\}]$*". This protocol generalized to several different groups, to representations, and to arbitrary modular relations.

(II) *Proof of knowledge of the discrete logarithm modulo a composite*: In [6,**?**], they apply such PK's to the group of quadratic residues modulo a composite n, $G = QR_n$. Thus the prover needs to convince the verifier that elements he presents are indeed quadratic residues. It is sufficient to execute $PK\{(\alpha) : y^2 = (g^2)^\alpha\}$ instead of $PK\{(\alpha) : y = g^\alpha\}$ [6]. The quantity α is defined as $\log_{g^2} y^2$ which is same as $\log_g y$ in case y is a quadratic residue.

We use the notation $PK^2\{(\alpha) : y = g^\alpha\}$ in the group of quadratic residues modulo a composite, simply.

3 Construction of Pseudonym System

3.1 Procedures

We give an overview of our pseudonym system in this section. The basic system comprises procedures, (1) *System setup*, (2) *Registration of an organization* (*Entry into the system of an organization*), (3) *Registration of a user* ((3-1) *Registration with CA* (*Entry into the system of a user*), (3-2) *Registration with a group*, (3-3) *Registration with an organization*), (4) *Proof the possession of a credential by a user* ((4-1) *Showing a credential with/without identity of an organization*, (4-2) *Transferring a credential with/without identity of an organization*). In our paper, throughout we assume that users, organizations and group managers are connected by perfect anonymous channels, and each protocol is executed through a secure channel.

1. System setup:

All group managers G_I generate their group secret and public-key pairs $(\mathcal{X}_{G_I}, \mathcal{Y}_{G_I})$ by running GK_{G_I}.

2. Registration with group $\boldsymbol{G}$ of organization O_i:
O_i runs $GK_{(O_i,G)}$, generates a secret and public-key pair $(\mathcal{X}_{(O_i,G)}, \mathcal{Y}_{(O_i,G)})$ by using $\boldsymbol{G}$'s public-key $\mathcal{Y}_G$, and registers $\mathcal{Y}_{(O_i,G)}$. A group manager G publishes a list of public-keys of organizations.

3-1. Registration with CA of user U:
After identification by U, CA checks that U is eligible to join the system. U generates a master secret-key $\mathcal{X}_U$ by running GK_U, both U and CA run $GP\langle U(\mathcal{X}_U), CA\rangle$ to establish U's pseudonym $\mathcal{P}_{(U,CA)}$ which is based on $\mathcal{X}_U$. Then U can receive a credential $\mathcal{C}_{(U,CA)}$, by running $IC\langle U, CA\rangle(\mathcal{P}_{(U,CA)})$.

3-2. Registration with group $\boldsymbol{G}$ of user U:
Both U and G run $GP\langle U(\mathcal{X}_U), G\rangle$ to establish U's pseudonym $\mathcal{P}_{(U,G)}$, and run $TC\langle U, G\rangle$ to demonstrate whether or not U is a valid participant in the system. In TC, U can prove the possession of $\mathcal{C}_{(U,CA)}$ on $\mathcal{P}_{(U,CA)}$ based on $\mathcal{X}_U$ where $\mathcal{P}_{(U,G)} \in GP\langle U(\mathcal{X}_U), G\rangle$. If it is valid, G issues a credential $\mathcal{C}_{(U,G)}$ on $\mathcal{P}_{(U,G)}$ to U by running $IC\langle U, G\rangle$.

3-3. Registration with organization $O_i \in \boldsymbol{G}$ of user U:
Both U and O_i run $GP\langle U(\mathcal{X}_U), O_i\rangle$ to get $\mathcal{P}_{(U,O_i)}$, and run $TC\langle U, O_i\rangle$ to prove the possession of $\mathcal{C}_{(U,G)}$ on $\mathcal{P}_{(U,G)}$ based on $\mathcal{X}_U$. If it is valid, then they run $GV\langle U, O_i\rangle$ to generate a proof of a validity of $\mathcal{P}_{(U,O_i)}$, whose output $\sigma_{(U,O_i)}$ guarantees that U has registered a pseudonym with O_i. Note that $\sigma_{(U,O_i)}$ is the U's private output. After G checks the validity of $\sigma_{(U,O_i)}$, G blindly issues a credential $\mathcal{C}_{(U,O_i)}$ on $\mathcal{P}_{(U,O_i)}$ by running $BIC\langle U(\mathcal{P}_{(U,O_i)}), G\rangle$.

4-1. Showing of a credential on a pseudonym with organization O_i: U chooses a way to show a credential. If U wants to let V know an organization $O_i \in \boldsymbol{G}$ with which U transacts, then both U and V run $SC^+\langle U, V\rangle(\mathcal{Y}_{(O_i,G)})$, which assures that U has $\mathcal{C}_{(U,O_i)}$ on $\mathcal{P}_{(U,O_i)}$ established with O_i. If U does not want to let V know the corresponding organization, both U and V run $SC^-\langle U(\mathcal{Y}_{(O_i,G)}), V\rangle$ which proves the only possession of a credential $\mathcal{C}_{(U,O_i)}$ on $\mathcal{P}_{(U,O_i)}$ without revealing $\mathcal{Y}_{(O_i,G)}, \mathcal{C}_{(U,O_i)}$ and $\mathcal{P}_{(U,O_i)}$.

4-2. Transferring a credential on a pseudonym with organization O_i: Let U register $\mathcal{P}_{(U,O_i)}$ and $\mathcal{P}_{(U,X_j)}$ with an organization $O_i \in \boldsymbol{G}_I$ and $X_j \in \boldsymbol{G}_J$ respectively. Both U and X_j execute $TC^+\langle U, X_j\rangle(\mathcal{Y}_{(O_i,G_I)})$ which assures that U has $\mathcal{C}_{(U,O_i)}$ on $\mathcal{P}_{(U,O_i)}$ based on $\mathcal{X}_U$ where $\mathcal{P}_{(U,X_j)} \in GP\langle U(\mathcal{X}_U), X_j\rangle$, without revealing $\mathcal{C}_{(U,O_i)}$ and $\mathcal{P}_{(U,O_i)}$. If U does not want to let X_j know the organization O_i, then both U and X_j run $TC^-\langle U(\mathcal{Y}_{(O_i,G_I)}), X_j\rangle$.

3.2 Constructions of Pseudonym Systems

This section provides constructions of our pseudonym system.

Common system parameter

Security-related system parameters are as follows: the length ℓ_n of the RSA modulus, the integer intervals $\Gamma =]-2^{\ell_\Gamma}, 2^{\ell_\Gamma}[$, $\Delta =]-2^{\ell_\Delta}, 2^{\ell_\Delta}[$, $\Lambda =]2^{\ell_\Lambda}, 2^{\ell_\Lambda+\ell_\Sigma}[$, such that $\ell_\Delta = \epsilon\ell_\Gamma$ and $\ell_\Gamma = 2\ell_n$, where $\epsilon > 1$ is a security parameter, and $2^{\ell_\Lambda} > 2(2^{2\ell_\Gamma} + 2^{\ell_\Gamma} + 2^{\ell_\Delta})$, and $2(2^{\ell_\Sigma}(2^{2\ell_\Gamma} + 2^{\ell_\Delta}) + 2^{\ell_\Delta}) < 2^{\ell_\Lambda}$.

Generation of keys

1. A group manager $G \in \boldsymbol{G}$ chooses random $\ell_n/2$-bit primes p'_G, q'_G such that $p_G := 2p'_G + 1$ and $q_G := 2q'_G + 1$ are prime, sets modulus $n_G := p_G q_G$. It also chooses elements $d_G, e_G, f_G, g_G, h_G \in_R QR_{n_G}$. It stores $\mathcal{X}_G := (p_G, q_G)$ as its secret-keys, and publishes $\mathcal{Y}_G := (n_G, d_G, e_G, f_G, g_G, h_G)$ as its public-key together with a proof that n_G is the product of two safe primes and that the elements d_G, e_G, f_G, g_G and h_G lie indeed in QR_{n_G}.
2. An organization O_i chooses a secret-key $x_{(O_i,G)} \in_R \Gamma$ and sets a corresponding public-key $y_{(O_i,G)} := {g_G}^{x_{(O_i,G)}} \pmod{n_G}$ to register with group $\boldsymbol{G}$. O_i stores $x_{(O_i,G)}$ as a secret-key $\mathcal{X}_{(O_i,G)}$ and publishes $y_{(O_i,G)}$ and its identity $id_{(O_i,G)}$ as O_i's public-keys $\mathcal{Y}_{(O_i,G)}$.
3. A user U chooses a random secret element $x_U \in \Gamma$, and stores it as U's master secret-key $\mathcal{X}_U$ in the system.

Generation of a pseudonym

$GP\langle U, X\rangle$ assures that $\mathcal{P}_{(U,X)} = (N_{(U,X)}, P_{(U,X)})$ is of right form, i.e., $P_{(U,X)} = {g_G}^{x_U} {h_G}^{s_{(U,X)}}$, with $x_U \in \Gamma$ and $s_{(U,X)} \in \Delta$. $N_{(U,X)}$ and $P_{(U,X)}$ are called a nym and its validating tag, respectively. To establish a pseudonym with an entity X, both U and X carry out the following protocol:

1. U chooses $N_1 \in \{0,1\}^k, r_1 \in_R \Delta$ and $r_2, r_3 \in_R \{0,1\}^{2\ell_n}$, sets $c_1 := {d_G}^{r_1} {e_G}^{r_2}$ and $c_2 := {d_G}^{x_U} {e_G}^{r_3}$. U sends N_1, c_1 and c_2 to X, and serves as the prover to verifier X in $PK^2\{(\alpha, \beta, \gamma, \delta) \ : c_1 = {d_G}^{\alpha} {e_G}^{\beta} \wedge c_2 = {d_G}^{\gamma} {e_G}^{\delta}\}$, to prove c_1 and c_2 are generated correctly.
2. X chooses $r \in_R \Delta$, and sends r and N_2 to U.
3. U sets the nym $N_{(U,X)} := N_1 || N_2$, and computes $s_{(U,X)} := (r_1 + r \pmod{2^{\ell_\Delta+1}+1}) - 2^{\ell_\Delta} + 1$, and $\tilde{s} = \lfloor (r_1 + r)/(2^{\ell_\Delta+1} - 1) \rfloor$. U sets $P_{(U,X)} := {g_G}^{x_U} {h_G}^{s_{(U,X)}}$ as a validating tag of $N_{(U,X)}$. U sends $P_{(U,X)}$ to X, and shows that it was formed correctly: U sets $c_3 := {d_G}^{\tilde{s}} {e_G}^{r_4}$ for $r_4 \in_R \{0,1\}^{\ell_n}$, sends it to X. Then they engage in

$$
\begin{aligned}
PK^2\{(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \vartheta, \xi) \ : \ & c_1 = {d_G}^{\alpha} {e_G}^{\beta} \\
& \wedge \ c_2 = {d_G}^{\gamma} {e_G}^{\delta} \\
& \wedge \ c_3 = {d_G}^{\varepsilon} {e_G}^{\zeta} \\
& \wedge \ P_{(U,X)} = {g_G}^{\gamma} {h_G}^{\vartheta} \\
& \wedge \ (c_1 (d_G)^{r - 2^{\ell_\Delta+1}}) / ({c_3}^{2^{\ell_\Delta+1}+1}) = {d_G}^{\vartheta} {e_G}^{\xi} \\
& \wedge \ \gamma \in \Gamma \wedge \vartheta \in \Delta\}.
\end{aligned}
$$

5. X stores $\mathcal{P}_{(U,X)} = (N_{(U,X)}, P_{(U,X)})$ in its database.
6. U stores $(\mathcal{S}_{(U,X)}, \mathcal{P}_{(U,X)}) = (s_{(U,X)}, \{N_{(U,X)}, P_{(U,X)}\})$ in its record with X.

Issue of a credential on a pseudonym with a group

$IC\langle U, G\rangle$ guarantees that a credential on a previously established $\mathcal{P}_{(U,G)}$ is $\mathcal{C}_{(U,G)} = (E_{(U,G)}, C_{(U,G)})$ such that $C_{(U,G)} \equiv (P_{(U,G)} f_G)^{1/E_{(U,G)}} \pmod{n_G}$. To be granted credential, U runs the following protocol with G:

1. U identifies as its owner by $PK^2\{(\alpha,\beta) : P_{(U,G)} = {g_G}^{\alpha}{h_G}^{\beta}\}$ for $\mathcal{P}_{(U,G)}$ in G's database.
2. G chooses a random prime $E_{(U,G)} \in_R \Lambda$, computes $C_{(U,G)} := (P_{(U,G)}f_G)^{1/E_{(U,G)}} \pmod{n_G}$, and sends $E_{(U,G)}$ and $C_{(U,G)}$ to U. Then G stores $\mathcal{C}_{(U,G)} = (E_{(U,G)}, C_{(U,G)})$ as a credential on $\mathcal{P}_{(U,G)}$.
3. U checks if ${C_{(U,G)}}^{E_{(U,G)}} \equiv P_{(U,G)}f_G \pmod{n_G}$ and $E_{(U,G)} \in \Lambda$, and stores $\mathcal{C}_{(U,G)} = (E_{(U,G)}, C_{(U,G)})$ in its record with group $\boldsymbol{G}$.

Showing a credential on a pseudonym with a group

U proves the possession of $\mathcal{C}_{(U,G)} \in IC\langle U, G\rangle$ by running SC. Both U and V engage in the following protocol:

1. U sets $c_1 := C_{(U,G)}{e_G}^{r_1}$ and $c_2 := {e_G}^{r_1}{d_G}^{r_2}$ for $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, and sends c_1 and c_2 to V,
2. U engages with V in

$$\begin{aligned} PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi) \,:\, & f_G = {c_1}^{\alpha}/{g_G}^{\beta}{h_G}^{\gamma}{e_G}^{\delta} \\ & \wedge\, c_2 = {e_G}^{\varepsilon}{d_G}^{\zeta} \\ & \wedge\, 1 = {c_2}^{\alpha}/{e_G}^{\delta}{d_G}^{\xi} \\ & \wedge\, \alpha \in \Lambda \wedge \beta \in \Gamma \wedge \gamma \in \Delta\}. \end{aligned}$$

Transferring a credential on a pseudonym with a group

TC assures that U owns $\mathcal{C}_{(U,G_I)}$ on $\mathcal{P}_{(U,G_I)}$ based on $\mathcal{X}_U$ where $\mathcal{P}_{(U,X_j)} \in GP\langle U(\mathcal{X}_U), X_j\rangle$. U proves it by running TC with $X_j \in \boldsymbol{G}_J$ with whom U has established $\mathcal{P}_{(U,X_j)}$:

1. U sets $c_1 := C_{(U,G_I)}{e_G}^{r_1}$ and $c_2 := {e_{G_I}}^{r_1}{d_{G_I}}^{r_2}$ for $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, and sends c_1 and c_2 to X_j,
2. U engages with X_i in

$$\begin{aligned} PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta) \,:\, & f_{G_I} = {c_1}^{\alpha}/{g_{G_I}}^{\beta}{h_{G_I}}^{\gamma}{e_{G_I}}^{\delta} \\ & \wedge\, c_2 = {e_{G_I}}^{\varepsilon}{d_{G_I}}^{\zeta} \\ & \wedge\, 1 = {c_2}^{\alpha}/{e_{G_I}}^{\delta}{d_{G_I}}^{\xi} \\ & \wedge\, P_{(U,X_j)} = {g_{G_J}}^{\beta}{h_{G_J}}^{\eta} \\ & \wedge\, \alpha \in \Lambda \wedge \beta \in \Gamma \wedge \gamma \in \Delta\}, \end{aligned}$$

for $\mathcal{P}_{(U,X_j)}$ in X_j's database.

Generation of a proof of pseudonym's validity

GV guarantees that U's output $\sigma_{(U,O_i)}$ is independent of O_i's view of the conversation. In order to generate a signature on $\mathcal{P}_{(U,O_i)}$, both U and O_i run GV:

1. U identifies as its owner by $PK^2\{(\alpha,\beta) : P_{(U,O_i)} = {g_G}^{\alpha}{h_G}^{\beta}\}$, for $\mathcal{P}_{(U,O_i)}$ in O_i's database.

2. O_i generates $Q_{(U,O_i)} := {P_{(U,O_i)}}^{x_{(O_i,G)}}$, $t_1 := {g_G}^r$ and $t_2 := {P_{(U,O_i)}}^r$ for $r \in_R \{0,1\}^{2\ell_n}$, and sends $Q_{(U,O_i)}, t_1$ and t_2 to U.

3. U chooses r_1, r_2 and $r_3 \in_R \{0,1\}^{2\ell_n}$ and computes ${t_1}' := t_1 {g_G}^{r_1} {y_{(O_i,G)}}^{r_2}$, ${t_2}' := (t_2 {P_{(U,O_i)}}^{r_1} {Q_{(U,O_i)}}^{r_2})^{r_3}$, $P'_{(U,O_i)} := {P_{(U,O_i)}}^{r_3}$ and $Q'_{(U,O_i)} := {Q_{(U,O_i)}}^{r_3}$. Then U sets $e' := \mathcal{H}(g_G, y_{(O_i,G)}, P'_{(U,O_i)}, Q'_{(U,O_i)}, {t_1}', {t_2}')$, sends $e := e' - r_2$ to O_i.

4. O_i computes $s := r - e x_{(O_i,G)}$ and sends it to U.

5. U checks if $t_1 = {g_G}^s {y_{(O_i,G)}}^e$, $t_2 = {P_{(U,O_i)}}^s {Q_{(U,O_i)}}^e$, and sets $s' := s + r_1$. Then U stores $\sigma_{(U,O_i)} := (e', s', P'_{(U,O_i)}, Q'_{(U,O_i)})$ as a proof of a validity of $\mathcal{P}_{(U,O_i)}$, and keeps r_3 secretly until U gets a credential on $\mathcal{P}_{(U,O_i)}$.

Issue of a credential on a pseudonym with an organization

$BIC\langle U, G\rangle$ guarantees that a credential on $\mathcal{P}_{(U,O_i)}$ is $\mathcal{C}_{(U,O_i)} = (E_{(U,O_i)}, C_{(U,O_i)})$ such that $C_{(U,O_i)} \equiv (P_{(U,O_i)} {d_G}^{id_{(O_i,G)}} f_G)^{1/E_{(U,O_i)}}$. BIC establishes $\mathcal{C}_{(U,O_i)}$ without revealing anything more than the fact that U has registered with O_i to G. Such a credential can be granted by using the blind RSA-signature [1] in the following protocol:

1. U chooses a prime $E_{(U,O_i)} \in_R \Lambda$ and $r \in_R \mathbb{Z}_{n_G}$, and generates $c := r^{E_{(U,O_i)}} P_{(U,O_i)}$ ${d_G}^{id_{(O_i,G)}} f_G$. Then U sends $c, E_{(U,O_i)}$ and $\sigma_{(U,O_i)}$. Furthermore U must show that $\sigma_{(U,O_i)}$ was generated to U and c was generated correctly: U computes $c_1 := r {e_G}^{r_1}$ for $r_1 \in_R \{0,1\}^{2\ell_n}$, and engages with G in

$$
\begin{aligned}
PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta) \,:\, & P_{(U,G)} = {g_G}^{\alpha} {h_G}^{\beta} \\
& \wedge\, 1 = {P_{(U,G)}}^{\gamma} / {g_G}^{\delta} {h_G}^{\varepsilon} \\
& \wedge\, P'_{(U,O_i)} = {g_G}^{\delta} {h_G}^{\zeta} \\
& \wedge\, P'_{(U,O_i)} = c^{\gamma} ({e_G}^{E_{(U,O_i)}})^{\xi} / ({c_1}^{E_{(U,O_i)}} {d_G}^{id_{(U,O_i)}} f_G)^{\gamma} \\
& \wedge\, \alpha \in \Gamma, \beta \in \Delta\},
\end{aligned}
$$

for $\mathcal{P}_{(U,G)}$ in G's database.

2. O_i checks if σ is valid: if $e' = \mathcal{H}(g_G, y_{(O_i,G)}, P'_{(U,O_i)}, Q'_{(U,O_i)}, \tilde{t_1}, \tilde{t_2})$ where $\tilde{t_1} = {g_G}^{s'} {y_{(O_i,G)}}^{e'}$, $\tilde{t_2} = {P'_{(U,O_i)}}^{s'} {Q'_{(U,O_i)}}^{e'}$, and $y_{(O_i,G)}$ is in $\boldsymbol{G}$'s public-key list. Then O_i computes $c' := c^{1/E_{(U,O_i)}}$ and sends it to U.

3. U sets $C_{(U,O_i)} := c'/r$. Then U checks if ${C_{(U,O_i)}}^{E_{(U,O_i)}} \equiv P_{(U,O_i)} {d_G}^{id_{(O_i,G)}} f_G \pmod{n_G}$, and stores $(E_{(U,O_i)}, C_{(U,O_i)})$ in its record with organization O_i.

Showing a credential with identity of an organization

To prove the possession of $\mathcal{C}_{(U,O_i)} \in BIC\langle U, G\rangle$, both U and V run SC^+. They engage in the following protocol:

1. U sets $c_1 := C_{(U,O_i)} {e_G}^{r_1}$ and $c_2 := {e_G}^{r_1} {d_G}^{r_2}$ for $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, and sends c_1 and c_2 to V,

2. U engages with V in

$$PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi) : f_G {d_G}^{id_{(O_i,G)}} = {c_1}^{\alpha}/{g_G}^{\beta}{h_G}^{\gamma}{e_G}^{\delta} \\ \wedge c_2 = {e_G}^{\varepsilon}{d_G}^{\zeta} \\ \wedge 1 = {c_2}^{\alpha}/{e_G}^{\delta}{d_G}^{\xi} \\ \wedge \alpha \in \Lambda \wedge \beta \in \Gamma \wedge \gamma \in \Delta\}.$$

Showing a credential without identity of an organization

In order to prove the possession of a credential generated by running $BIC\langle U, G\rangle$, both U and V run SC^-. They engage in the following protocol:

1. U sets $c_1 := C_{(U,O_i)}{e_G}^{r_1}$ and $c_2 := {e_G}^{r_1}{d_G}^{r_2}$ for $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, and sends c_1 and c_2 to V,
2. U engages with V in

$$PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta) : f_G = {c_1}^{\alpha}/{g_G}^{\beta}{h_G}^{\gamma}{d_G}^{\delta}{e_G}^{\varepsilon} \\ \wedge c_2 = {e_G}^{\zeta}{d_G}^{\xi} \\ \wedge 1 = {c_2}^{\alpha}/{e_G}^{\varepsilon}{d_G}^{\eta} \\ \wedge \alpha \in \Lambda \wedge \beta \in \Gamma \wedge \gamma \in \Delta\}.$$

Transferring a credential with identity of an organization

In TC^+, U proves the possession of $\mathcal{C}_{(U,O_i)}$ on $\mathcal{P}_{(U,O_i)}$ based on $\mathcal{X}_U$ where $\mathcal{P}_{(U,X_j)} \in GP\langle U(\mathcal{X}_U), X_j\rangle$ to $X_j \in \boldsymbol{G}_J$:

1. U sets $c_1 := C_{(U,O_i)}{e_{G_I}}^{r_1}$ and $c_2 := {e_{G_I}}^{r_1}{d_{G_I}}^{r_2}$ for $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, and sends c_1 and c_2 to X_j,
2. U engages with X_j in

$$PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta) : {f_{G_I}}{d_{G_I}}^{id_{(O_i,G_I)}} = {c_1}^{\alpha}/{g_{G_I}}^{\beta}{h_{G_I}}^{\gamma}{e_{G_I}}^{\delta} \\ \wedge c_2 = {e_{G_I}}^{\varepsilon}{d_{G_I}}^{\zeta} \\ \wedge 1 = {c_2}^{\alpha}/{e_{G_I}}^{\delta}{d_{G_I}}^{\xi} \\ \wedge P_{(U,X_j)} = {g_{G_J}}^{\beta}{h_{G_J}}^{\eta} \\ \wedge \alpha \in \Lambda \wedge \beta \in \Gamma \wedge \gamma \in \Delta\},$$

for $\mathcal{P}_{(U,X_j)}$ in X_j's database.

Transferring a credential without identity of an organization

U proves the possession of a credential generated by running $BIC\langle U, G_I\rangle$ on a pseudonym based on $\mathcal{X}_U$ where $\mathcal{P}_{(U,X_j)} \in GP\langle U(\mathcal{X}_U), X_j\rangle$:

1. U sets $c_1 := C_{(U,O_i)}{e_{G_I}}^{r_1}$ and $c_2 := {e_{G_I}}^{r_1}{d_{G_I}}^{r_2}$ for $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, and sends c_1 and c_2 to X_j,

2. U engages with X_j in

$$PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta,\varphi) : f_{G_I} = {c_1}^{\alpha}/{g_{G_I}}^{\beta}{h_{G_I}}^{\gamma}{d_{G_I}}^{\delta}{e_{G_I}}^{\varepsilon} \wedge c_2 = {e_{G_I}}^{\zeta}{d_{G_I}}^{\xi} \wedge 1 = {c_2}^{\alpha}/{e_{G_I}}^{\varepsilon}{d_{G_I}}^{\eta} \wedge P_{(U,X_j)} = {g_{G_J}}^{\beta}{h_{G_J}}^{\varphi} \wedge \alpha \in \Lambda \wedge \beta \in \Gamma \wedge \gamma \in \Delta\},$$

for $P_{(U,X_j)}$ in X_j's database.

4 Proof of Security for Our Scheme

In this section, we assess the security of our pseudonym system. Under the strong RSA assumption and the decisional Diffie-Hellman assumption modulo a safe prime product, the following technical lemmas about the protocols described are stated here:

Lemma 3 *The* $PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\vartheta,\xi) : c_1 = {d_G}^{\alpha}{e_G}^{\beta} \wedge c_2 = {d_G}^{\gamma}{e_G}^{\delta} \wedge c_3 = {d_G}^{\varepsilon}{e_G}^{\zeta} \wedge P_{(U,X)} = {g_G}^{\gamma}{h_G}^{\vartheta} \wedge (c_1(d_G)^{r-2^{\ell_\Delta+1}})/({c_3}^{2^{\ell_\Delta+1}+1}) = {d_G}^{\vartheta}{e_G}^{\xi} \wedge \gamma \in \Gamma \wedge \vartheta \in \Delta\}$ *in GP is a statistical zero-knowledge proof of knowledge of the correctly formed values* $x_U, s_{(U,X)}$ *that correspond to a pseudonym validating tag* $P_{(U,X)}$.

Lemma 4 *The* PK^2 *protocols in* SC, TC, SC^+, SC^-, TC^+ *and* TC^- *are a statistical zero-knowledge proof of knowledge of the prover's master secret-key, corresponding secret information(s) and credential in right form.*

The proofs of these Lemmas is closely related to Lemma 1, 2 and 3 of [6], we only prove the security of the PK^2 protocol in TC^- here. The other proofs can easily inferred from the following.

Lemma 5 *The* $PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta,\varphi) : f_{G_I} = {c_1}^{\alpha}/{g_{G_I}}^{\beta}{h_{G_I}}^{\gamma}{d_{G_I}}^{\delta}{e_{G_I}}^{\varepsilon} \wedge c_2 = {e_{G_I}}^{\zeta}{d_{G_I}}^{\xi} \wedge 1 = {c_2}^{\alpha}/{e_{G_I}}^{\varepsilon}{d_{G_I}}^{\eta} \wedge P_{(U,X_j)} = {g_{G_J}}^{\beta}{h_{G_J}}^{\varphi} \wedge \alpha \in \Lambda \wedge \beta \in \Gamma \wedge \gamma \in \Delta\}$ *in* TC^- *is a statistical zero-knowledge proof of knowledge of the values* $x \in \Gamma, s_1, s_2 \in \Delta, E \in \Lambda, C$, *and* y *such that* $P_{(U,X_j)} = {g_{G_J}}^{x}{h_{G_J}}^{s_1} \pmod{n_{G_J}}$, *and* $C^E = {g_{G_I}}^{x}{h_{G_I}}^{s_2}{d_{G_I}}^{y}f_{G_I} \pmod{n_{G_I}}$.

Proof. By the properties of the PK^2 and the RSA assumption, the knowledge extractor can produce values $\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta,\varphi$ such that the statement after the colon holds. As $c_2 = {e_{G_I}}^{\zeta}{d_{G_I}}^{\xi}$ and $1 = {c_2}^{\alpha}/{e_{G_I}}^{\varepsilon}{d_{G_I}}^{\eta}$ from which we conclude that $\zeta\alpha = \varepsilon \pmod{ord(e_{G_I})}$, we have ${c_1}^{\alpha}/{e_{G_I}}^{\varepsilon} = {g_{G_I}}^{\beta}{h_{G_I}}^{\gamma}{d_{G_I}}^{\delta}f_G = (c_1/{e_{G_I}}^{\zeta})^{\alpha}$, where $\alpha \in \Lambda, \beta \in \Gamma$ and $\gamma \in \Delta$. It follows that U must know a valid credential $c_1/{e_{G_I}}^{\zeta}$ on a pseudonym. Furthermore, from $P_{(U,X_j)} = {g_{G_J}}^{\beta}{h_{G_J}}^{\eta}$, it guarantees that both pseudonym $P_{(U,X_j)}$ and a pseudonym registered with O_i are based on the same master secret key.

Lemma 6 *The* $PK^2\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta): P_{(U,G)}={g_G}^{\alpha}{h_G}^{\beta} \wedge 1=P_{(U,G)}{}^{\gamma}/{g_G}^{\delta}{h_G}^{\varepsilon} \wedge P'_{(U,O_i)} = {g_G}^{\delta}{h_G}^{\zeta} \wedge P'_{(U,O_i)} = c^{\gamma}({e_G}^{E_{(U,O_i)}})^{\xi}/({c_1}^{E_{(U,O_i)}}{d_G}^{id_{(U,O_i)}}f_G)^{\gamma} \wedge \alpha \in \Gamma, \beta \in \Delta\}$ *in BIC is a statistical zero-knowledge proof of knowledge of the values* $x \in \Gamma, s \in \Delta, r$, *and* r_3 *such that* $P_{(U,G)} = {g_G}^{x}{h_G}^{s}$, $c = r^{E_{(U,O_i)}}P_{(U,O_i)}{d_G}^{id_{(O_i,G)}}f_G$ *and* $P'_{(U,O_i)} = {P_{(U,O_i)}}^{r_3}$.

Proof. In the statement after the colon, $P_{(U,G)}={g_G}^{\alpha}{h_G}^{\beta}$ and $1=P_{(U,G)}{}^{\gamma}/{g_G}^{\delta}{h_G}^{\varepsilon}$ from which we conclude $\alpha\gamma \equiv \delta \pmod{ord(g_G)}$. From $P'_{(U,O_i)} = c^{\gamma}({e_G}^{E_{(U,O_i)}})^{\xi}/({c_1}^{E_{(U,O_i)}}{d_G}^{id_{(U,O_i)}}f_G)^{\gamma}$, we have $c^{\gamma} = {g_G}^{\delta}{h_G}^{\zeta}({c_1}^{E_{(U,O_i)}}{d_G}^{id_{(O_i,G)}}f_G)^{\gamma}/({e_G}^{E_{(U,O_i)}})^{\xi} = \{(c_1/{e_G}^{\xi/\gamma})^{E_{(U,O_i)}}{g_G}^{\alpha}{h_G}^{\zeta/\gamma}{d_G}^{id_{(O_i,G)}}f_G\}^{\gamma}$. As $\alpha \in \Gamma$ and $\beta \in \Delta$, c is formed collectly, by using the same key underlying $P_{(U,G)}$.

4.1 Description of the Simulator

We have to describe simulator $\mathcal{S}$ for our scheme and then show that it satisfies *Definition 1*. The parties the adversary $\mathcal{A}$ controls are subsumed into a single party. We only describe the simulator for the adversary.

Setup.

For the group manager $G \in \boldsymbol{G}$ and the organization $O \in \boldsymbol{G}$ not controlled by $\mathcal{A}$, $\mathcal{S}$ sets up their secret and public-key $(\mathcal{X}_G, \mathcal{Y}_G)$, and $(\mathcal{X}_{(O_i,G)}, \mathcal{Y}_{(O_i,G)})$ as dictated by the protocol. Furthermore, $\mathcal{S}$ creates an $archive_G$ or $archive_O$ where it will record the credentials of users controlled by $\mathcal{A}$ with the group manager or the organization. It also initialized a list of users controlled by $\mathcal{A}$, $list_{\mathcal{A}}$.

Generation of a pseudonym with a group.

A user establishes a pseudonym with a group manager G:

(I) If a user is controlled by $\mathcal{A}$, (i) $\mathcal{S}$ uses the knowledge extractor of Lemma 3 to discover the user's master secret key x and the secret values s, (i-1) If $x \notin list_{\mathcal{A}}$, $\mathcal{S}$ creates a new user U with login name L_U, and obtains a pseudonym $N_{(U,G)}$, and a key K_U corresponding to L_U by interaction with T. Denote (x, s) by $(x_U, s_{(U,G)})$, $\mathcal{S}$ stores $(U, L_U, x_U, K_U, N_{(U,G)}, s_{(U,G)})$ in $list_{\mathcal{A}}$, (ii-2) If $x \in list_{\mathcal{A}}$, $\mathcal{S}$ obtains $N_{(U,G)}$ for this user U corresponding to x by interaction with T, and adds $N_{(U,G)}, s_{(U,G)} := s$ to U's record.

(II) If a group manager G is controlled by $\mathcal{A}$, $\mathcal{S}$ will use the zero-knowledge simulator from Lemma 3 to furnish the $\mathcal{A}$'s view of the protocol.

Issue a credential on a pseudonym with a group.

A user requests a group manager G to issue a credential:

(I) If a user is controlled by $\mathcal{A}$, (i) upon receiving a message from T, $\mathcal{S}$ runs the knowledge extractor for the proof of knowledge of step 1 of IC, to determine the value x and s. For N corresponding to (x, s), (i-1) if $N \notin list_{\mathcal{A}}$, then $\mathcal{S}$ refuses to grant a credential. (i-2) If $N \in list_{\mathcal{A}}$, then $\mathcal{S}$ issued the correct E and C by interaction with T. $\mathcal{S}$ stores the values $(x_U, s_{(U,G)}, E_{(U,G)}, C_{(U,G)}) := (x, s, E, C)$ in $archive_G$.

(II) If a group manager G is controlled by $\mathcal{A}$, $\mathcal{S}$ will run the zero-knowledge simulator for step 1 of IC, and continue the protocol as U would. If the user accepts, then $\mathcal{S}$ informs T that the credential was granted.

Generation of a pseudonym with an organization.

A user establishes a pseudonym with an organization $O \in \boldsymbol{G}$:

This part of the simulator can easily be inferred from the part for the above *Generation of a pseudonym with a group.*

Generation of a proof of pseudonym's validity.

A user requests an organization O to grant a proof of pseudonym's validity: (I) If a user is controlled by $\mathcal{A}$, (i) $\mathcal{S}$ uses the knowledge extractor for PK of step 1 of GV to discover the user's key x and the value s. For N corresponding to (x, s), (i-1) If $N \notin list_{\mathcal{A}}$, $\mathcal{S}$ refuses to grant a proof of pseudonym's validity. (i-2) If $N \in list_{\mathcal{A}}$, $\mathcal{S}$ grants σ by interaction with T.

(II) If an organization O is controlled by $\mathcal{A}$, $\mathcal{S}$ will run the zero-knowledge simulator for step 1 of GV, and continue the protocol as U would.

Issue a credential on a pseudonym with an organization.

A user requests a group manager $G \in \boldsymbol{G}$ to issue a credential with an organization $O \in \boldsymbol{G}$:

(I) If a user is controlled by $\mathcal{A}$, (i) upon receiving a message from T, $\mathcal{S}$ runs the knowledge extractor for the proof of knowledge of step 1 of BIC to extract the value $x, s_{(U,G)}, s_{(U,O)}$ and r. (i-1) If $(x, s_{(U,G)}) \notin archive_G$, then $\mathcal{S}$ refuses to grant a credential, (i-2) If $(x, s_{(U,G)}) \in archive_G$, then $\mathcal{S}$ issues the correct c' corresponding to E by executing the rest of the G's side of it. $\mathcal{S}$ determines C by c'/r. It stores the values $(x, s_{(U,O)}, C, E)$ in $archive_O$.

(II) If a user is controlled by $\mathcal{A}$ and an organization $O \in \boldsymbol{G}$ is dishonest, (i) upon receiving a message from T, $\mathcal{S}$ runs the knowledge extractor for the proof of knowledge of step 1 of BIC, to extract the value $x, s_{(U,G)}, s$ and r. $\mathcal{S}$ looks at $archive_O$: (i-1) If $(x, s) \in archive_O$, $\mathcal{S}$ denotes this user by U, (i-2) If $(x, s) \notin archive_O$, let U be the user with x. $\mathcal{S}$ obtains $N_{(U,O)}$ by interaction with T, (ii) $\mathcal{S}$ looks at $archive_G$: (ii-1) If $(x, s_{(U,G)}) \notin archive_G$, then $\mathcal{S}$ refuses to grant a credential, (ii-2) If $(x, s_{(U,G)}) \in archive_G$, then $\mathcal{S}$ issues the correct c' corresponding to E by executing the rest of the G's side of it. $\mathcal{S}$ determines C by c'/r. It stores the values $(x, s_{(U,O)}, C, E)$ in $archive_O$.

(III) If an issuing group manager $G \in \boldsymbol{G}$ controlled by $\mathcal{A}$, $\mathcal{S}$ will run the zero-knowledge simulator for step 1 of BIC, and execute the rest of the user's side of it. If the user accepts, then $\mathcal{S}$ informs T that the credential was granted.

Showing a credential with identity of an organization

Showing a credential without identity of an organization

Transferring a credential with identity of an organization

These parts of the simulator can easily be inferred from the part for *Transferring a credential without identity of an organization* that follows.

Transferring a credential without identity of an organization.

A user wants to show ownership of a credential of a pseudonym with some organization in a group $\boldsymbol{G}_I$ to an organization $O_j \in \boldsymbol{G}_J$:

(I) If a user is controlled by $\mathcal{A}$, (i) $\mathcal{S}$ runs O_j's part of TC^-, and extracts the values $x, s_{(U,O_i)}, s_{(U,O_j)}, E, C$ and $y_{(O_i,G_I)}$ with the knowledge extractor of Lemma 5. (i-1) if $(x, s_{(U,O_i)}, E, C) \notin archive_{O_i}$, $\mathcal{S}$ rejects, (i-2) If $(x, s_{(U,O_i)}, E, C) \in archive_{O_i}$, $\mathcal{S}$ communicates with T for transferring a credential by U.

(II) If a user is controlled by $\mathcal{A}$ and an issuing group manager G_I is dishonest, (i) $\mathcal{S}$ runs O_j' side of CT^- with the knowledge extractor of Lemma 5 to obtain the values $x, s, s_{(U,O_j)}, E, C$ and y, let O_i be an organization whose public-key is y. (i-1) If O_j's side of the protocol reject, it does nothing, (i-2) Otherwise: (2-A-a) If $x \in archive_{O_i}$, denote this user by U, (2-A-b) If $x \notin archive_{O_i}$, let U be the user with x, and $\mathcal{S}$ obtain $N_{(U,O_i)}$ by interaction with T. (2-B) If $(E, C) \notin archive_{O_i}$, then $\mathcal{S}$ runs BIC, adds the output to U's record. (2-C) $\mathcal{S}$ communicates with T for transferring a credential by U. (III) If a verification organization O_j is controlled by $\mathcal{A}$, $\mathcal{S}$ runs the zero-knowledge simulator of Lemma 5 to do that.

4.2 Proof of Successful Simulation

We show that our simulator fails with negligible probability only. We show in the following lemma that a tuple (x, s, E, C) the knowledge of which is essential for proving possession of a credential, is unforgeable even under an adaptive attack. As these proofs can be found in [6], we leave out the proofs.

Lemma 7 *Under the strong RSA assumption and the discrete logarithm assumption modulo a safe prime product, if a polynomially bounded adversary succeeds in proving ownership of a valid credential record (P, E, C) with a group $\boldsymbol{G}$, then this credential record was created by running GP, IC and TC with a group manager $G \in \boldsymbol{G}$.*

Lemma 8 *Under the strong RSA assumption, the discrete logarithm assumption modulo a safe prime product and , if a polynomially bounded adversary succeeds in proving ownership of a valid credential record (P, E, C) with an organization $O \in \boldsymbol{G}$, then this credential record was created by running GP and TC with an organization $O \in \boldsymbol{G}$, BIC with a group manager $G \in \boldsymbol{G}$.*

The statistical zero-knowledge property of the underlying protocols gives us Lemma 9 which in turn implies Theorem 10.

Lemma 9 *The view of the adversary in the real protocol is statistically close to his view in the simulation.*

Theorem 10 *Under the strong RSA assumption, the decisional Diffie-Hellman assumption modulo a safe prime product, and the assumption that factoring is hard, our pseudonym system described above is secure.*

5 Conclusion

This paper presents an anonymity-enhanced pseudonym system; a user can select a way to prove the possession of a credential on a pseudonym with an organization. We can add a mechanism: *global anonymity revocation* for discovering the identity of a user whose transactions are illegal, or *local anonymity revocation* for revealing a pseudonym of a user who misuses the credential, in the same way as [6].

References

1. D.Chaum, "Security without identification: Transaction systems to make big brother obsolete", Communications of the ACM, vol. 28, 1030–1044, 1985
2. D.Chaum and J.-H.Evertse, "A secure and privacy - protecting protocol for transmitting personal information between organizations", Proceedings of CRYPTO'86, vol. 263, 118–167, Springer Verlag, 1987
3. L.Chen, "Access with pseudonyms", Cryptography: Policy and Algorithms, vol. 1029, 232–243, Springer Verlag, 1995
4. I.B.Damgard, "Payment systems and credential mechanism with provable security against abuse by individuals", Proceedings of CRYPTO'88, vol. 403, 328–335, Springer Verlag, 1990
5. A.Lysyanskaya and R.Rivest and A.Sahai and S.Wolf, "Pseudonym Systems", Selected Areas in Cryptography, vol. 1758, Springer Verlag, 1999
6. J.Camenisch and A.Lysyanskaya, "Efficient non-transferable anonymous multishow credential system with optional anonymity revocation", Proceedings of EUROCRYPT 2001, vol. 2045, 93–118, Springer Verlag, 2001
7. J.Camenisch and A.Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials", Proceedings of CRYPTO 2002, vol. 2442, 61–76, Springer Verlag, 2002
8. J.Camenisch and E.V.Herreweghen, "Design and implementation of the idemix anonymous credential system", ACM CCS'02, 2002
9. G.Ateniese and J.Camenisch and M.Joye and G.Tsudik, "A practical and provably secure coalition-resistant group signature scheme", Proceedings of CRYPTO 2000, vol. 1880, 255–270, Springer Verlag, 2000
10. C.P.Schnorr, "Efficient signature generation for smart cards", Journal of Cryptology, vol. 4, 239–252, 1991
11. A.Fiat and A.Shamir, "How to prove yourself: Practical solution to identification and signature problems", Proceedings of CRYPTO'86, vol. 263, 186–194, Springer Verlag, 1987
12. E.Fujisaki and T.Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations", Proceedings of CRYPTO'97, vol. 1294, 16–30, Springer Verlag, 1997
13. J.Camenisch and M.Stadler, "Efficient group signature schemes for large groups", Proceedings of CRYPTO'97, vol. 1294, 410–424, Springer Verlag, 1997
14. R.Cramer and V.Shoup, "Signature schemes based on the strong RSA assumption", Proceedings of 6th ACM Conference on Computer and Communications Security, 46–52, ACM press, 1999
15. M.Bellare and C.Namprempre and D.Pointcheval and M.Semanko, "The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme", Proceedings of Financial Cryptography 2001, vol. 2339, 319–338, Springer Verlag, 2001

Using Feedback to Improve Masquerade Detection

Kwong H. Yung

Stanford University Computer Science Department
353 Serra Mall; Stanford CA 94305-9025; USA
khyung@cs.stanford.edu

Abstract. To gain access to account privileges, an intruder masquerades as the proper account user. Information from user feedback helps to improve the accuracy of classifiers used for detecting masquerades. Instead of operating in isolation, the online sequential classifier can request feedback from the user. In the full-feedback policy, the classifier verifies every session; in the feedback-on-alarm policy, the classifier confirms only suspicious sessions. Surprisingly, confirming only a few sessions under the feedback-on-alarm policy is enough to be competitive with verifying all sessions under the full-feedback policy. Experiments on a standard artificial dataset demonstrate that the naive-Bayes classifier boosted by the feedback-on-alarm policy beats the previous best-performing detector and reduces the number of missing alarms by 30%.

Keywords: feedback-on-alarm, feedback policy; sequential classifier, online classifier; naive-Bayes classifier, adaptive classifier; masquerading session, masquerading user; masquerade detection, intrusion detection.

1 Introduction

This paper presents a simple technique for identifying masquerading sessions in a multi-user system. Profiles of proper and intruding behaviors are built on the sessions of known users. As new, unidentified sessions are presented, the classifier identifies the sessions and then has the option of requesting feedback confirmation from the user. In this online sequential classification context, the proposed *feedback-on-alarm* policy markedly improves on earlier detection rates.

1.1 Motivation

After penetrating a computer network, a computer hacker seeks to break into available user accounts. Upon breaking into a user account, the computer hacker masquerading as the proper user can gain access to privileges reserved for the proper user of the account. Unless the intruder's activities can be detected as a masquerade, the intruder can perform malicious attacks under the guise of the proper user.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 48–62, 2003.

In most modern multi-user computer systems, individual user accounts are typically identified by a username and protected by only a password. With the correct username and password combination, anyone can fool the computer system into granting account privileges specifically reserved to the proper user.

1.2 Elementary Solutions

To prevent unauthorized users, computer systems can be equipped with additional safeguards beyond the usual combination of username and password. Potentially, biometric technology will become powerful enough to verify the identify of the proper user, by measuring unique bodily characteristics such as fingerprints, retina, or voice. Moreover, multiple physical barriers can be erected to keep potential intruders away from physical access to the underlying computer system.

Yet these additional safeguards often fail to prevent attacks from insiders. Because many computer crimes are committed by insiders, these safeguards can become useless. According to a recent survey [9], insider employees were the culprits in an estimated 70% of computer-systems intrusions leading to losses. In the most extreme case, even the *proper* user can use his account for deviant activity. For example, a security officer may abuse his account privileges to download secret files for illicit purposes. In a recent widely publicized case [4], an FBI agent of the United States served as a spy for Russia and used his own account to obtain top-secret information.

Even as computer technology advances and modern systems become more secure, the problem of masquerading users remains a crucial hole in computer security. A determined human intruder often preys upon security leaks and upon errors in human behavior. Powerful as these technologies may be, cryptography, biometrics, and other security mechanisms, often fail to undermine the main leak in computer security: the human factor.

1.3 Previous Approaches

The artificial dataset used in this study has been studied by many researchers [1, 2,3,5,7,8]. Subsection 4.1 presents a summary of the dataset, and full details are available in [7]. Before this paper, the best result was achieved by using a naive-Bayes classifier with updating [5].

Schonlau and his colleagues [7] compared six different techniques for detecting masquerading sessions. The review paper cited two techniques based on statistical inference, two techniques from the machine-learning literature, and two techniques based on heuristics.

In the two techniques of classical statistical inference, the command sequence was modeled as a Markov chain, and a variety of hypothesis-testing procedures were applied. The results were disappointing, suggesting that typical user command sequences cannot be modeled adequately by a Markov chain. The Markov assumption also presented many technical difficulties, because there were not

enough data to estimate the many parameters required by the model. A sequence of earlier technical reports [1,2,3,8], presenting several variations on the Markov model for command sequences, were also disappointing.

Later, Maxion and Townsend [5] used a naive-Bayes classifier, which uniformly outperformed the six techniques summarized by Schonlau and his colleagues [7]. Maxion and Townsend formulated the anomaly-detection problem as a classification problem by comparing each user against the other 49 users. As a simplification, each session was reduced to a histogram, and the ordering of the commands was ignored. Because each session was treated as a bag of words, the problem became amenable to familiar techniques of text classification. At 0.013 false-alarm rate, the naive-Bayes classifier had a missing-alarm rate of only 0.385, a one-third decrease from the previous best result cited in [7].

1.4 Early Limitations

Both Maxion and Townsend [5] and Schonlau and his colleagues [7] tried to correct for the non-stationarity of user behavior by updating user profiles. In [7], detectors with updating had fewer false alarms and generally outperformed the versions without updating. In [5], the naive-Bayes classifier with updating performed uniformly better than the naive-Bayes classifier without updating, by a wide margin.

The results of [5] and [7] demonstrate clearly that updating the detector produced significant improvements by lowering the number of false alarms. As new sessions are presented to the detector, the detector must classify the new sessions and update profiles. In both [5] and [7], the detector considers each new session separately in sequence and must decide immediately how to classify the new session. Once the detector decides that the new session is proper, the detector then updates the profile of the proper user to include the new session, as if the new session had been part of the training set.

There are at least two limitations to the updating scheme used in [5] and [7]. Principally, the detector must make a concrete decision whether to update the proper profile with the new session. In many cases, the identity of a session is not clear. Yet the detector is forced to make a binary decision before the detector can continue, because future decisions depend on the decision for the current session.

Furthermore, the detector suffers from its own mistakes. If the detector mistakes a masquerading session for a proper session, then after updating, the masquerading session will contaminate the profile of the proper user. Because false alarms are quite common, the detector often leaves out a proper session from the proper profile and potentially makes the same error on future sessions. In both cases, an error made by the detector will be propagated onto future decisions. Therefore, early errors are compounded at each stage and become quite costly as time progresses.

1.5 New Strategy

Because human behavior is complicated and non-stationary, predicting changes in behavior is nearly impossible. The updating procedure used in [5] and [7] becomes impractical on users with erratic changes in behavior. In fact, any detector built strictly on examples of past behavior is fundamentally limited because concept drift is hardly predictable on the basis of old data.

In the online context, however, more information can be requested as soon as concept drift is detected. Borrowing a chapter from active learning [6] in classification, this paper proposes to use feedback from the user in the online-classification problem. The detector can ask questions to prevent mistakes as well as to improve future decisions. Naturally investigating the true identity of a session is costly. Yet not soliciting feedback can result in far more costly penalties, in the number of future false alarms and missing alarms.

This paper tests two simple policies for requesting feedback: feedback-on-alarm and full feedback. Under the feedback-on-alarm policy, the detector asks for confirmation after detecting a suspicious session. Under the full-feedback policy, the detector asks for confirmation after each and every session. This full-feedback policy is not practical but represents a theoretical limit, against which the feedback-on-alarm policy can be compared.

The two simple feedback policies presented in this paper are applied to the the naive-Bayes classifier used in [5]. As expected, feedback improves detection rate. More surprisingly, the feedback-on-alarm policy performs almost as well as the full-feedback policy. Apparently, the detector can dramatically improve its accuracy by verifying the identities of a few carefully chosen sessions.

1.6 Outline of Paper

Following this introduction, Section 2 provides a brief background on the standard naive-Bayes classifier used for detecting masquerading sessions. Section 3 then formulates the strategy of using feedback in a more general setting. Section 4 presents results of experiments on a standard artificial dataset. Section 5 discusses the advantages of the feedback-on-alarm policy as well as potential for future work. Finally, Section 6 summarizes this paper's main conclusions.

2 Background

The feedback-on-alarm policy proposed in this paper can be applied to any detector operating in an online sequential context. To test the improvements offered by this feedback policy, the base case will be the current best performing classifier, naive-Bayes with updating but without feedback [5].

Although the feedback-on-alarm policy is a significant step beyond the naive-Bayes classifier used by Maxion and Townsend [5], three elements introduced there still apply here: the classification paradigm, the bag-of-words model, and the naive-Bayes classifier. These three strategies are not unique to detecting

masquerading sessions, but rather they appear in the far more general context of intrusion detection and text classification. Below, a brief review of the three elements appears in the context of detecting masquerading sessions.

2.1 Classification Paradigm

As in much of intrusion detection, examples of normal behavior are plentiful, but intrusions themselves are rare. In the context of detecting masquerading sessions, many examples of proper sessions are available. In the training data, however, no masquerading sessions are known. Moreover, masquerading sessions in the test data are expected to be rare.

Naturally, proper sessions in the training data are used to build the profile of proper sessions. To create a profile for masquerading sessions, the sessions of *other* known users in the training data are used. For the dataset of [7], this approach is particularly appropriate, because the artificially created masquerading sessions do indeed come from known users excluded from the training set.

Each new session is compared against the profile of the proper sessions and the profile of the masquerading sessions. The anomaly-detection problem has been reformulated as a classification problem over the proper class and the artificially created intruder class. This reformulation allows many powerful techniques of classification theory to be used on the anomaly-detection problem.

Even in a more general context of anomaly detection, the profile of the proper sessions is defined not only by the proper sessions but also by the sessions of other users. The extent of the proper profile in feature space is most naturally defined through both the proper cases and the intruder cases. So the classification paradigm is potentially useful even for other anomaly-detection problems.

2.2 Bag-of-Words Model

The so-called bag-of-words model is perhaps the most widely used model for documents in information retrieval. A text document can be reduced to a bag of words, by ignoring the sequence information and only counting the number of occurrences of each word. For classification of text documents, this simple model often performs better than more complicated models involving sequence information.

Let $\mathbf{C} = \{1, 2, \ldots, C\}$ be the set of all possible commands. User session number s is simply a finite sequence $c_s = (c_{s1}, c_{s2}, \ldots, c_{sk}, \ldots, c_{sz_s})$ of commands. In other words, session number s of length z_s is identified with a sequence $c_s \in \mathbf{C}^{z_s}$. In the bag-of-words model, the sequence information is ignored. So c_s is reduced from the sequence (c_{sk}) to the multi-set $\{c_{sk}\}$.

As demonstrated in [5], the bag-of-words model outperforms many more complicated models attempting to take advantage of the sequence information. In particular, the techniques reviewed in [7] and earlier techniques based on a Markov model of command sequences achieved worse results.

2.3 Naive-Bayes Classifier

In the field of text classification, the simple naive-Bayes classifier is perhaps the most widely studied classifier, used in conjunction with the bag-of-words model for documents. The naive-Bayes classifier is the Bayes-rule classifier for the bag-of-words model.

Suppose that each user has a distinct pattern of command usage. Let c_s denote the sequence of commands in session number s. By Bayes rule, the probability $P(u|c_s)$ that the sequence c_s was generated by user u is

$$P(u|c_s) = \frac{P(c_s|u)P(u)}{P(c_s)} \propto P(c_s|u)P(u), \tag{1}$$

where $P(u)$ is the prior probability for user u. In practice, c_s is assigned to the user $u_0 = \text{argmax}\,\{P(c_s|u)P(u)|u = 1, 2, \ldots, U\}$, among U different users. In other words, the session c_s is assigned to the user u_o who most likely generated that session.

In the naive-Bayes model, each command is assumed to be chosen independently of the other commands. User u has probability p_{uc} of choosing command c. Because each command is chosen independently, the probability $P(c_s|u)$, that user u produced the sequence $c_s = (c_{s1}, c_{s2}, \ldots, c_{sk}, \ldots, c_{sz_s})$ of commands in session number s, is simply

$$P(c_s|u) = \prod_{k=1}^{z_s} P(c_{sk}|u) = \prod_{k=1}^{z_s} p_{uc_{sk}} = \prod_{c=1}^{C} p_{uc}^{n_{sc}}, \tag{2}$$

where $n_{sc} = \sum_{k=1}^{z_s} 1_{\{c_{sk}=c\}}$ is the total count of command c in session s.

3 Theory

The simple naive-Bayes classifier is typically built on a training set of labeled instances. Maxion and Townsend [5] demonstrated that classification accuracy can be improved significantly by updating the classifier with newly classified instances from the test set. The feedback-on-alarm policy in this paper extends that updating process by incorporating information from the user. At the same time, the theory explaining the benefits of feedback can also be used to justify the success of the previous adaptive-without-feedback policy.

This section continues the development of the probabilistic model introduced in Section 2. First, the maximum-likelihood formalism is used to estimate the model parameters necessary for implementing the naive-Bayes classifier. Then shrinkage estimators are introduced to account for rare commands. The effect of feedback is explained relative to the likelihood model.

3.1 Maximum-Likelihood Estimation

In Section 2, the bag-of-words model for documents was introduced. The session was defined to be a sequence of independent and identically distributed multivariate Bernoulli random variables over the set of possible commands. Then the

general Bayes rule for classification was reduced to the naive-Bayes classification rule. In practice, the parameters of the probabilistic model must be estimated from the data itself.

From now on, only two distinct classes of sessions are considered, the proper class and the intruder class. The indicator *variable* $1_s = 1$ exactly when session s is a masquerading session. Let $1 - \epsilon$ and ϵ be the prior probabilities that a session is proper and masquerading, respectively; let p_c and p'_c be the probability of command c in a proper and masquerading session, respectively.

The log-likelihood L_s of session s is simply, up to an additive constant

$$L_s = (1 - 1_s)(\log(1 - \epsilon) + \sum_{c=1}^{C} n_{sc} \log p_c) + 1_s(\log \epsilon + \sum_{c=1}^{C} n_{sc} \log p'_c), \quad (3)$$

where n_{sc} is the total count of command c in session s.

Assuming that all sessions are generated independently of each other, the cumulative log-likelihood L^t after t sessions is, up to an additive constant

$$L^t = \sum_{s=1}^{t} L_s = w^t \log(1 - \epsilon) + \sum_{c=1}^{C} n^t_{+c} \log p_c + w'^t \log \epsilon + \sum_{c=1}^{C} n'^t_{+c} \log p'_c, \quad (4)$$

where

$$w^t = \sum_{s=1}^{t} (1 - 1_s), w'^t = \sum_{s=1}^{t} 1_s, n^t_{+c} = \sum_{s=1}^{t} (1 - 1_s) n_{sc}, n'^t_{+c} = \sum_{s=1}^{t} 1_s n_{sc}. \quad (5)$$

Here w^t and $w'^t = t - w^t$ are the cumulative numbers of proper and masquerading sessions, respectively; n^t_{+c} and n'^t_{+c} are the cumulative counts of command c amongst proper sessions and masquerading sessions, respectively, in the t total observed sessions.

3.2 Shrinkage Estimators

Rare classes and rare commands may not be properly reflected in the training set. To avoid zero estimates, smoothing is typically applied to the maximum-likelihood estimators. This smoothing can also be motivated by shrinkage estimation under Dirichlet priors on the parameters ϵ, p_c, and p'_c.

Suppose that $p \sim \text{Dirichlet}(\alpha)$, $p' \sim \text{Dirichlet}(\alpha')$, and $(1-\epsilon, \epsilon) \sim \text{Beta}(\beta, \beta')$. Then the revised cumulative log-likelihood L is, up to an additive constant

$$L^t = (w^t + \beta - 1) \log(1 - \epsilon) + \sum_{c=1}^{C} (n^t_{+c} + \alpha_c - 1) \log p_c +$$
$$(w'^t + \beta' - 1) \log \epsilon + \sum_{c=1}^{C} (n'^t_{+c} + \alpha'_c - 1) \log p'_c. \quad (6)$$

So after t total sessions, the cumulative shrinkage estimator $\hat{\epsilon}^t$ for ϵ is

$$\hat{\epsilon}^t = \frac{w'^t + \beta' - 1}{t + \beta - 1 + \beta' - 1}. \tag{7}$$

The cumulative shrinkage estimators $\hat{p}_c^t$ and $\hat{p}_c'^t$ for p_c and p'_c after t sessions are

$$\hat{p}_c^t = \frac{n_{+c}^t + \alpha_c - 1}{\sum_{d=1}^{C}(n_{+d}^t + \alpha_d - 1)}, \hat{p}_c'^t = \frac{n'^t_{+c} + \alpha'_c - 1}{\sum_{d=1}^{C}(n'^t_{+d} + \alpha'_d - 1)}. \tag{8}$$

3.3 Updating Naive-Bayes Classifier

A new *unidentified* session t is suspicious if $P(1_t = 0|c_t; \hat{\theta}^{t-1})$ is too small, where $\theta = (\epsilon, p, p')$ denotes the model parameters. Equivalently from Bayes rule Equation 1 on this two-class problem, the classifier under threshold h will flag session t as a masquerade if

$$\log \frac{P(1_t = 0|c_t; \hat{\theta}^{t-1})}{P(1_t = 1|c_t; \hat{\theta}^{t-1})} = \log \frac{1 - \hat{\epsilon}^{t-1}}{\hat{\epsilon}^t} + \sum_{c=1}^{C} n_{tc} \log \frac{\hat{p}_c^{t-1}}{\hat{p}_c'^{t-1}} < h. \tag{9}$$

After test session t is classified, $1_t = 0$ or $1_t = 1$ is fixed. The updated model $\hat{\theta}^t = (\hat{\epsilon}^t, \hat{p}^t, \hat{p}'^t)$ is calculated using Equations 7 and 8 and is then used to classify the next test session $t + 1$.

3.4 Updating without Feedback

Initially, the training set is used to build the naive-Bayes classifier. As a new unidentified session is presented to the classifier, that new session is scored by the classifier and used to update the classifier.

For a session s in the training set, the identity is known. Specifically, $1_s = 0$ for a proper session, and $1_s = 1$ for a masquerading session in the training set. For a new unidentified session, 1_s is a missing variable that can be estimated from the available data. For added assurance, however, feedback from the user provides confirmation.

In Maxion and Townsend [5], the adaptive-without-feedback policy improved remarkably on the frozen-profile policy. Feedback is used to verify the true values of 1_s for unidentified sessions. In the absence of feedback, however, the classifier can make its best guess on 1_s. If the classifier guesses 1_s correctly, then it is rewarded; otherwise, the classifier is penalized later. As long as the classifier has reliable accuracy, it can trust its own guesses even without feedback confirmation. Naturally feedback provides more assurance and improves even poorly performing classifiers.

- **Inputs:**
  - Target user u and cut-off threshold h for deciding new sessions.
  - Initial training set $\mathbf{R}$ of identified sessions from all users.
  - Sequence $\mathbf{S}$ of new unidentified sessions, purportedly from the target user u.
- Build initial classifier for user u on training set $\mathbf{R}$. Use Equations 7 and 8 to calculate initial model $\hat{\theta}^0 = (\hat{\epsilon}^0, \hat{p}^0, \hat{p}'^0)$ from training set $\mathbf{R}$.
- Loop over each new session $t = 1, 2, \ldots, S$ in test sequence $\mathbf{S}$:
  - Raise alarm $1_t = 1$ if Equation 9 holds.
  - Evaluate decision 1_t against actual identity of session t.
  - Apply the feedback policy to correct 1_t if necessary.
  - Use Equations 7 and 8 to calculate updated model $\hat{\theta}^t$.
- **Outputs:**
  - Classifier model $\hat{\theta}^S$ for target user u built on training set $\mathbf{R}$ and updated by the sequence $\mathbf{S}$ of sessions, under input threshold h and feedback policy.
  - Online scores of sessions in sequence $\mathbf{S}$, before feedback.
  - False alarms and missing alarms under sequential evaluation on $\mathbf{S}$.

Fig. 1. Sequential evaluation of feedback policy.

3.5 Sequential Evaluation

The algorithm for online classification of new sessions uses sequential presentation, sequential scoring, and sequential evaluation. For a fair comparison between classifiers with and without feedback, all classifiers are evaluated sequentially, on its initial predictions *before* the feedback is made available.

Figure 1 shows the algorithm used for sequential evaluation of online classifiers with feedback. The online classifier is presented with a sequence of new sessions, one-by-one. The classifier also scores each sessions sequentially. Moreover, the classifier is evaluated against its predictions in a sequential manner. For classifier using feedback, the classifier is still penalized by its initial scores. However, feedback allows the classifier to correct itself *before* updating its profiles for the next session.

4 Results

The feedback policies are tested on a standard dataset previously analyzed by other authors using different techniques. The dataset contains command logs from 50 users on a UNIX multi-user system. For each of the 50 users, there is a training portion without masquerading sessions and a test portion with possible masquerading sessions from the artificially created intruders. To create artificial intrusions, sessions from a separate group of 20 users are injected at random into the command logs of the 50 known users.

First a brief overview of the reference dataset is provided below, but full details are provided in [7]. Then the experiments performed are described in detail. Finally, the results are compared with [5].

4.1 Reference Dataset

For privacy reasons, the command options and arguments were deleted from the collected logs, leaving only the commands themselves. For each of 70 users, the first 15000 recorded commands were kept. Then 50 users were chosen at random to serve as the dataset, and the remaining 20 users were excluded, except to provide artificial intrusion data. For each user, the sequence of 15000 commands was divided into 150 sequential blocks, each containing 100 commands in sequence. Each artificially created block is treated as a session. Therefore, each of the 50 users had 150 sessions, each of 100 commands.

The first 50 sessions of the 50 users are declared to be free of intruders. These 50 sessions for the 50 users constitute the training set. The remaining 100 sessions for each of the 50 users form the test set. Sessions from the 20 excluded users were injected into the test portions of the 50 known users. The extra sessions in the test portions are then removed, leaving just 100 test sessions for each of the 50 known users.

To prevent over-training on these command logs, the following details of the data creation were not used as part of the classification procedure proposed in this paper. Essentially, once an intruder has broken into a user account, he is likely to return on the very next session. A proper session is followed by a masquerading session with 0.01 probability. A masquerading session is followed by a masquerading session from the same intruder with 0.80 probability. A chain of masquerading sessions is taken from that intruder's log in a contiguous sequence.

4.2 Experimental Design

The details for one single experiment are described below. The example shows how masquerading sessions on User 12 were detected. For other users, the same analysis applies.

The full training set was divided into two classes. The proper class consisted of 50 sessions for User 12. The intruder class consisted of the 50 sessions for each of the other 49 users. So the intruder class was built on a total of 2450 sessions.

The test set for user 12 had 100 sessions, all purportedly proper sessions. The classifier for user 12 was then run against each of the 100 sessions in a sequential fashion, as detailed in Figure 1. The Dirichlet parameters $\alpha_c - 1 = 0.01$ and $\alpha'_c - 1 = 0.01$ for all $c \in \mathbf{C}$ were chosen to match the parameters used in [5]. Also to match [5], the fraction ϵ was held fixed in the model tested in this paper, and consequently the ϵ term could be excluded from Equation 9. Equivalently, session t was flagged as a masquerade under some threshold k when

$$\sum_{c=1}^{C} n_{tc} \log \frac{\hat{p}_c^{t-1}}{\hat{p}_c'^{t-1}} < k. \tag{10}$$

4.3 Composite ROC Curves

A separate classifier was created for each user and used to score that user's test sessions. The scores of different test sessions for one single user can be

compared because those scores were assigned by that user's classifier. However, scores from different users cannot easily be compared because those scores were assigned from different classifiers with different scoring systems.

To evaluate a policy's success over the entire dataset of 50 users, a composite ROC curve can be useful. There are several common ways to combine the individual 50 ROC curves into one single curve, but all methods are at best arbitrary.

In this paper, all 50 classifiers were run on a common set of threshold cut-offs. The composite ROC curve was also plotted over the same set of thresholds. At each threshold, the false alarms were summed over all users. Similarly, at each threshold, the missing alarms were summed over all users.

4.4 Experimental Results

The naive-Bayes classifier was built for each of the 50 users. Essentially 50 separate tests were conducted, one for each user. Rather than report the results for each user separately, composite results involving all 50 users are reported instead.

Under the full-feedback policy, the number of questions asked is equal to the number of sessions in the sequence. Under the feedback-on-alarm policy, the number of questions asked is equal to the total number of alarms raised, the number of false alarms plus the number of true alarms.

The ROC curves were not adjusted in any way to account for the cost of feedback. Instead, the algorithm in Figure 1 evaluated each decision in sequence before feedback. Feedback was incorporated only after the classifier was scored against its initial decision, and so the classifier may be penalized before it uses feedback to change its decision. This sequential evaluation procedure placed policies with feedback on equal ground with policies without feedback.

4.5 Boost from Feedback

Figure 2 shows the composite ROC curves of all 50 users, for false-alarm rate in the 0.00–0.10 range. There are a total of 231 masquerading sessions and 4769 proper sessions. Four policies are compared: frozen profile, adaptive without feedback, adaptive with feedback on alarm, and adaptive with full feedback. As expected, policies with feedback uniformly outperform policies without feedback. Surprisingly, the feedback-on-alarm policy performs almost as well as the full-feedback policy.

Indeed, on the naive-Bayes classifier, the feedback-on-alarm policy outperforms uniformly the adaptive-without-feedback policy. In particular, for false-alarm rate 0.002, the feedback-on-alarm policy reduces the missing-alarm rate of the adaptive-without-feedback policy by 30%. To achieve this reduction, the feedback-on-alarm policy asks about $0.002(4769) + 0.40(231) = 102$ questions total: 10 false alarms and 92 true alarms.

The ROC curves show that the incremental gains diminish in the progression from the frozen-profile policy to the full-feedback policy. In changing from a

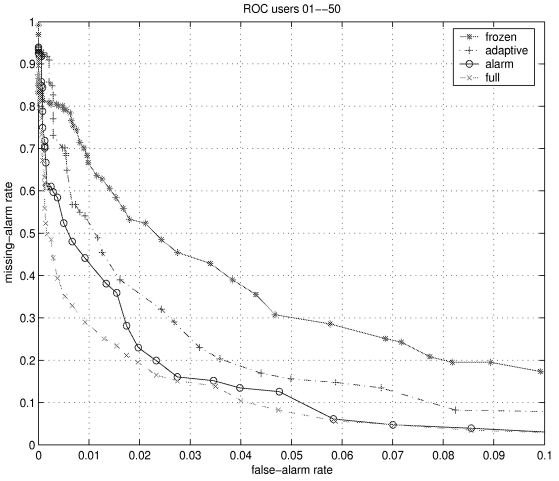

Fig. 2. ROC curve composite over users 01–50.

frozen-profile policy to an adaptive-without-feedback policy, the classifier can make primitive corrections for concept drift. In moving from adaptive-without-feedback policy to the feedback-on-alarm policy, the classifier can correct itself on false alarms. In moving from the feedback-on-alarm policy to the full-feedback policy, the classifier can correct itself also on missing alarms.

Figure 3 shows the ROC curve for user 12 alone. On user 12, the adaptive-without-feedback policy performs worse than random guessing. Apparently, the adaptive-without-feedback policy updates the proper profile with a few masquerading sessions. Under the feedback-on-alarm policy, however, the improvements are dramatic. In fact, the 50 ROC curves for each individual user typically show that the feedback-on-alarm policy improves over the adaptive-without-feedback by a significant margin.

5 Discussion

Applying feedback to the naive-Bayes classifier significantly improved detection rates, over all settings. For false-alarm rate 0.002, the naive-Bayes classifier with feedback-on-alarm lowers the missing-alarm rate by 30%. The feedback-on-alarm policy performed almost as well as the full-feedback policy. Thus, the feedback-on-alarm policy is both useful and practical in most realistic contexts.

5.1 Large-Scale Scenario

The feedback-on-alarm policy prevents contamination of profiles at a small burden to the user. This advantage will become more dramatic as the number of users and the number of sessions increase. As the classifier learns from more

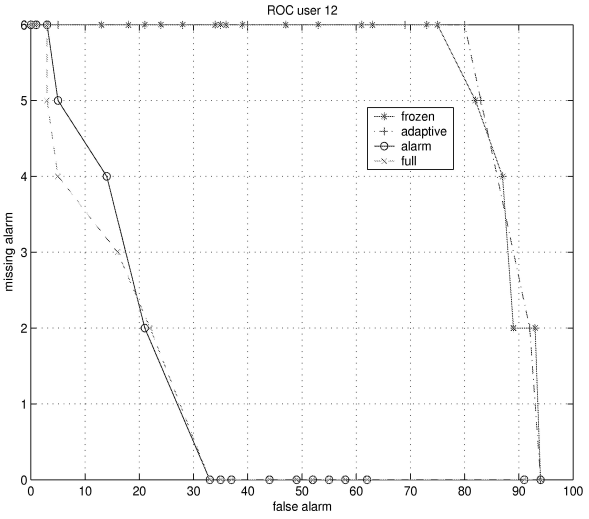

Fig. 3. ROC curve user 12.

cases, mistakes from earlier decisions are propagated onwards and magnified. Without feedback, the errors are like to be compounded repeated at each successive stage. In a large-scale scenario, the improvements from the feedback-on-alarm policy over the adaptive-without-feedback policy are expected to become far more pronounced.

5.2 Long-Term Concept Drift

The naive-Bayes classifier relies only on command frequencies. As a user's behavior changes over time, the profiles built from past sessions become outdated. An exponential-weighting process can be applied to counts from past sessions. For the naive-Bayes classifier, this re-weighting of counts is especially simple. Such an extension becomes even more valuable in realistic scenarios, where an online classifier is used over an extended period of time.

5.3 Future Directions

Only the simplest feedback policies have been presented in this paper. In principle, sounding an alarm and asking for feedback are two separate decisions, based on two different criteria. More information may be gained from borderline cases than from extreme cases. Choosing the set of sessions on which to request feedback can be posed as an optimization problem under the likelihood model.

In the context of detecting masquerades, however, most users do not want be faced with seemingly unnecessary questions. In practice, an alarm would trigger an investigation, and so implementing the feedback-on-alarm policy incurs

no additional cost. Moreover, this paper demonstrates that requesting feedback on suspicious sessions is near optimal, performing almost as well as asking for feedback on all sessions.

6 Summary

In previous approaches for detecting masquerades, the detector was forced to make a binary decision about the identity of a new session. Without any assurance on its decision, a classifier may update its profiles incorrectly and suffer severe penalties at later stages. Under the feedback-on-alarm policy, the online sequential classifier can confirm the identities of suspicious sessions. Experiments on the naive-Bayes classifier show that the feedback-on-alarm policy reduces the number of missing alarms from the adaptive-without-feedback policy by 30%.

The simple feedback-on-alarm policy has two advantages. Primarily, it corrects false alarms, common in masquerade detection. Moreover, because in a real-world scenario these alarms would trigger investigations, the feedback-on-alarm policy incurs no additional cost. Compared to the full-feedback policy, the feedback-on-alarm policy achieves competitive results, with far fewer questions. Essentially, the full-feedback policy corrects only a few missing alarms left undetected by the feedback-on-alarm policy.

In principle, requesting feedback and raising an alarm are two separate decisions. Yet because the simple feedback-on-alarm policy performs remarkably near the theoretical optimal limit, the feedback-on-alarm policy is a natural choice for practical implementations, even if more complicated feedback policies can offer potentially better detection.

Acknowledgments. This research project was funded in part by the US Department of Justice grant 2000-DT-CX-K001. Jerome H. Friedman of the Stanford University Statistics Department provided invaluable feedback through many helpful discussions. Jeffrey D. Ullman of the Stanford University Computer Science Department introduced the author to the field of intrusion detection and offered insightful critiques throughout the past three years. The author is grateful for their advice and support.

References

1. William DuMouchel. “Computer intrusion detection based on Bayes factors for comparing command transition probabilities.” Technical Report 91, National Institute of Statistical Sciences, Research Triangle Park, North Carolina 27709–4006, 1999.
2. William DuMouchel and Matthias Schonlau. “A comparison of test statistics for computer intrusion detection based on principal components regression of transition probabilities.” *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics*, 30:404–413, 1999.

3. Wen-Hua Ju and Yehuda Vardi. "A hybrid high-order Markov chain model for computer intrusion detection." Technical Report 92, National Institute for Statistical Sciences, Research Triangle Park, North Carolina 27709–4006, 1999.
4. Vernon Loeb. "Spy case prompts computer search." *Washington Post*, page A01, 05 March 2001.
5. Roy A. Maxion and Tahlia N. Townsend. "Masquerade detection using truncated command lines." *International Conference on Dependable Systems and Networks (DSN-02)*, pp. 219–228, Washington, DC, 23–26 June 2002. IEEE Computer Society Press, Los Alamitos, California, 2002.
6. Andrew K. McCallum and Kamal Nigam. "Employing EM in pool-based active learning for text classification." *Machine Learning: Proceedings of the Fifteenth International Conference (ICML '98)*, pp. 350–358, 1998.
7. Matthias Schonlau, William DuMouchel, Wen-Hua Ju, Alan F. Karr, Martin Theus, and Yehuda Vardi. "Computer intrusion: detecting masquerades." *Statistical Science*, 16(1):58–74, February 2001.
8. Matthias Schonlau and Martin Theus. "Detecting masquerades in intrusion detection based on unpopular commands." *Information Processing Letters*, 76(1–2):33–38, November 2000.
9. Bob Tedeschi. "E-commerce report: crime is soaring in cyberspace, but many companies keep it quiet." *The New York Times*, page C4 column 1, 27 January 2003.

Efficient Presentation of Multivariate Audit Data for Intrusion Detection of Web-Based Internet Services

Zhi Guo[1], Kwok-Yan Lam[1], Siu-Leung Chung[2], Ming Gu[1], and Jia-Guang Sun[1]

[1] School of Software, Tsinghua University, Beijing, PR China
{gz, lamky, guming, sunjg}@tsinghua.edu.cn
[2] School of Business Administration, The Open University of Hong Kong
slchung@ouhk.edu.hk

Abstract. This paper presents an efficient implementation technique for presenting multivariate audit data needed by statistical-based intrusion detection systems. Multivariate data analysis is an important tool in statistical intrusion detection systems. Typically, multivariate statistical intrusion detection systems require visualization of the multivariate audit data in order to facilitate close inspection by security administrators during profile creation and intrusion alerts. However, when applying these intrusion detection schemes to web-based Internet applications, the space complexity of the visualization process is usually prohibiting due to the large number of resources managed by the web server. In order for the approach to be adopted effectively in practice, this paper presents an efficient technique that allows manipulation and visualization of a large amount of multivariate data. Experimental results show that our technique greatly reduces the space requirement of the visualization process, thus allowing the approach to be adopted for monitoring web-based Internet applications.

Keywords: Network security, Intrusion detection, Multivariate data analysis, Data visualization

1 Introduction

Effective security measures for protecting web-based services are important to organizations providing Internet applications. With the pervasive use of the Internet as a channel for delivering services, distributed applications are often implemented as Internet services over the web architecture. Internet applications usually adopt the 3-tier architecture consisting of a database layer, an application server layer and a web server layer. It is the nature of Internet applications that web servers are open to access from the global Internet. Thus web servers are exposed to serious security threats, and computer crimes targeting at web sites have become increasingly ubiquitous due to vulnerabilities of web servers.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 63–75, 2003.

Security monitoring is an important requirement in the protection of computer systems against attackers and malicious users. According to a CSI/FBI Computer Crime and Security Survey [1], 90% of large corporations and government agencies in the US detect computer security breaches and 80% acknowledged financial losses due to these breaches. Hence, for organizations with mission-critical Internet applications, such as financial institutions that provide e-financial services through the Internet, the immediate alert of and investigation into attacking and malicious activities are of vital importance to avoid asset losses as well as reputation losses.

Intrusion detection is an effective way to provide security monitoring function. The idea of intrusion detection [2,3] is based on the hypothesis that computers are typically involved in specific types of activity, and the set of programs and commands they execute will normally reflect that activity. Hence, security violations can be detected from abnormal patterns of system use in that exploitation of a system's vulnerabilities involves abnormal use of the system.

Intrusion detection systems typically consist of two major system components, the intrusion detector and the system administration console, as depicted in Figure 1. The intrusion detector compares ongoing activities with established behavior profile and detects abnormal activities by identifying deviation of these ongoing usage patterns from the established behavior profile. Behavior profiles are established based on audit records of activities collected under normal usage conditions. Should abnormal behavior be detected, an intrusion alert is raised and detailed information about the suspicious activities is presented to the security administrator at the system administration console. The administrator then investigates the information closely and decides whether the alert should be escalated to a higher level in accordance with the organization's security policy.

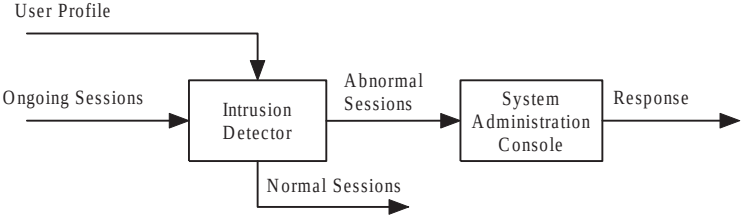

Fig. 1. The intrusion detection scheme.

Statistical-based analysis is an important area in intrusion detection [4,5,6, 7]. These techniques need to have the capability of analyzing a huge volume of multivariate data. Intrusion detection techniques typically involve the manipulation and analysis of audit data. Audit data of computer systems are mostly

multivariate in nature (e.g. measures of set of commands executed or web resources accessed by users). Furthermore, for web-based systems, audit data are voluminous. For example, the web-based digital library service at our university serves over 100,000 clicks per day with over 2,700 different types of web resources being accessed, thus generating a huge amount of audit data to be processed by the intrusion detection system.

A profiling stage that aims to create behavior profiles of the target system is important to the effectiveness of intrusion detection techniques. During the profiling stage, the target system is monitored over a period of time with audit logs being collected and analyzed. The historical activity logs of the system are summarized in form of behavior profiles of the target system. Depending on the underlying hypothesis, the profiles may be created based on different statistical approaches such as correlation analysis or other statistical measures [5,6,7]. The security administrator may closely inspect and fine-tune the profiles. In this connection, human inspection of the audit data is typically required. It is a basic cognitive nature of human being that visual display of complex information is usually more comprehensible than other form of information. Thus, efficient techniques for visualizing the multivariate audit data play an important role in intrusion detection.

In general, the system administration console is the human-machine interface of the intrusion detection system that allows the system administrator to perform visual inspection of audit data. In practice, this system administration console is most likely a machine with limited computing resources such as a high-end PC. Since the audit data to be inspected are voluminous, it may not be feasible to perform computations and manipulation of the multivariate data on such machines. Hence an efficient technique that allows manipulation and visualization of a large amount of multivariate data is needed.

This paper presents an efficient implementation technique for presenting multivariate audit data needed by statistical-based intrusion detection systems. The rest of the paper is organized as follows: Section 2 reviews an important technique for analyzing and visualizing multivariate audit data for intrusion detection purposes. The space requirement of the visualization technique becomes prohibiting when processing audit logs collected on web-based Internet applications. A new data visualization approach with significant improvement in space requirement is presented in Section 3. Experimental results that show the effectiveness of the approach are presented in Section 4. This paper is concluded by Section 5.

2 Multivariate Statistical Analysis of Audit Trails

Multivariate data analysis has been adopted successfully to detect misuses of computer systems [5,6,7]. Depending on the underlying intrusion detection hypothesis, different multivariate statistical analysis such as correlation analysis or

other statistical measures may be applied to analyze audit logs. For instance, the correlation analysis approach is based on the hypothesis that users involving in specific types of applications will execute commands that are strongly correlated. This is especially true in web-based applications where web resources are usually correlated due to the applications' design nature; e.g. multi-frame pages will cause multiple web resources to be accessed simultaneously.

The multivariate data analysis technique is used to create behavior profiles of intrusion detection systems. For example, correlation analysis helps select independent intrusion detection measures [5]. It may also be used to detect anomaly activities by analyzing correlation between user sessions [7]. In this case, intrusion detection is based on the hypothesis that activities of a normal user are strongly correlated, thus leading to the formation of a normal behavior profile. Ongoing user activities are compared against the normal behavior profile. Basically, the correlation between ongoing user activities and the user profile is computed. Ongoing activities that are strongly correlated with the profile are considered normal.

There are three steps in the multivariate data analysis intrusion detection model to detect misuses in web applications. Firstly, intrusion detection measures are extracted from the web server's audit log. The extracted data are represented in multivariate data form. Secondly, correlation analysis on the measures is performed. Thirdly, visualization technique is performed to allow the analysis results to be closely inspected by the administrator.

2.1 Data Representation

Before multivariate data analysis can be performed to detect intrusion, intrusion detection measures on which the intrusion detection model is based need to be defined. In the context of intrusion detection of web-based applications, it is the measures of web resources within user sessions that constitute the multivariate data to be analyzed. In our study, the frequency of access to each web resource is used as an intrusion detection measure.

With user session and intrusion detection measures defined, we can represent the data for multivariate data analysis in the form of a matrix M as follow:

$$
\begin{array}{cc}
 & \text{Measures} \\
 & \begin{array}{cccccc} 1 & 2 & \cdots & j & \cdots & p \end{array} \\
\text{Sessions} & \begin{array}{c|cccccc}
1 & & & & . & & \\
2 & & & & . & & \\
\vdots & & & & & & \\
i & & & \cdots & x_{ij} & & \\
\vdots & & & & & & \\
n & & & & & &
\end{array}
\end{array}
$$

where x_{ij} represents the frequency count of access to web resource j $(1 \leq j \leq p)$ in session i $(1 \leq i \leq n)$, i.e. the frequency count of accessing the j^{th} web resources in the i^{th} time interval (session). An example of M is shown in Figure 2.

	/index.htm	/bookview/index.htm	...	/Java/jv30.htm	/Java/jv31.htm
2003/2/20 8:00	5	6		0	0
2003/2/20 8:01	3	1		0	0
...					
2003/2/27 17:42	7	8		2	0
2003/2/27 17:43	12	6		0	1

Fig. 2. An example of M.

Geometrically, the matrix M can be viewed as consists of n row vectors, $\overrightarrow{X}_1, \cdots, \overrightarrow{X}_n$, in a p-dimensional (p-D) Euclidean space or p column vectors, $\overrightarrow{Y}_1, \cdots, \overrightarrow{Y}_p$, in an n-D Euclidean space. The n^{th} row vectors represent to n^{th} sessions and the p^{th} column vectors represent the p^{th} measures in the web log.

2.2 Correlation Analysis

As mentioned, an effective way to detect misuses is to test for correlation between ongoing user sessions and the behavior profile. Mathematically, a straightforward way is to compute and analyze the coefficient of correlation r_{ij} between user sessions such that

$$r_{ij} = \frac{1}{p} \sum_{k=1}^{p} \frac{(x_{ik} - \overline{x}_i)(x_{jk} - \overline{x}_j)}{\sigma_i \sigma_j}, \quad 1 \leq i, j \leq n \tag{1}$$

where

$$\overline{x}_i = \frac{\sum_{j=1}^{p} x_{ij}}{p} \quad , \quad 1 \leq i \leq n \tag{2}$$

is the mean of the vector of row i and

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{p} (x_{ij} - \overline{x}_i)^2}{p}} \quad , \quad 1 \leq i \leq n \tag{3}$$

is the standard deviation of the vector in row i.

However, it would be inefficient to compute the pair-wise coefficient of correlation for all the row vectors in M. Instead, the matrix is transformed so that correlated sessions can be identified by a cluster recognition approach using, for example, a minimum spanning tree (MST) algorithm. The transformation involves normalization of the row vectors in M and is described in the following:

$$\widetilde{x}_{ij} = \frac{(x_{ij} - \overline{x}_i)}{\sigma_i \sqrt{p}} \quad 1 \leq i \leq n, 1 \leq j \leq p. \tag{4}$$

Let the normalized matrix be $\widetilde{M}$, and let the i^{th} normalized row vector be $\widetilde{X}_i$. Geometrically, for any two normalized row vectors $\widetilde{X}_i$ and $\widetilde{X}_j$, which represent two points in p-D Euclidean space, their distance satisfies the following:

$$d^2(\widetilde{X}_i, \widetilde{X}_j) = 2(1 - r_{ij}) \tag{5}$$

where r_{ij} is the coefficient of correlation between $\widetilde{X}_i$ and $\widetilde{X}_j$. Thus, points which are close to each other are closely correlated (with a relatively small r_{ij}). In this respect, to recognize groups of closely correlated vectors (sessions), a cluster recognition technique can be used.

2.3 Visualization of Multivariate Data

Data visualization technique enables human inspection of the distribution of audit data. For example, abnormal activities can be identified visually if observed data deviate from the normal user profile. Since the number of measures in the audit data is very large, it is necessary to project points in high dimensional space to 2-D (or 3-D) space to allow visualization.

Different dimensionality reduction techniques based on unsupervised data, include factor analysis with principal components analysis (PCA), multidimensional scaling (MDS), Sammon's mapping (SAM), Kohonen's self-organizing maps (SOM) and auto-associative feedforward neural networks (AFN), and so on, are widely adopted to extract the major features of a data pattern in image processing [8,9]. In our study, we adopt factor analysis [10] which, as suggested in [7], projects high dimensional audit data to 2-D or 3-D space and at the same time preserve the major characteristics of the data pattern.

Factor analysis for dimensionality reduction is described as follows:

1. Compute $\widetilde{M}$ with row vectors equal the normalized row vectors of M.
2. Compute $\widetilde{M}^T \widetilde{M}$.
3. Compute the p eigenvalues $\lambda_1, \cdots, \lambda_p$ of $\widetilde{M}^T \widetilde{M}$ and the corresponding p orthonormal eigenvectors $\overrightarrow{u}_1, \cdots, \overrightarrow{u}_p$.
4. Project the n normalized points (i.e. the n normalized row vectors in $\widetilde{M}$) in p-D space to 2-D space with coordinates $(\widetilde{X}_i^T \overrightarrow{u}_1, \widetilde{X}_i^T \overrightarrow{u}_2)$, $i = 1 \ldots n$.

As mentioned in the previous section, audit data visualization is an important step in intrusion detection in that a more accurate estimation of the normal user profile can be achieved. In this respect, the volume of audit data that need to be processed and visualized is inevitably huge in size. For example, audit data extracted from the web log of our digital library are collected for a period of one week, the corresponding matrix M consists of about 600,000 rows (i.e.

sessions). In this case, M is huge and the space complexity of factor analysis is very high. Computing resources may not be enough to make the intrusion detection processes infeasible.

In the next section, a new approach for improving the space complexity of the data visualization processed is presented.

3 The Efficient Data Visualization Technique

A major obstacle to visualization of audit data for intrusion detection is the high space requirement in the matrix computation of the normalized matrix $\widetilde{M}$. In order to reduce the space complexity, a new approach is proposed for the data visualization process. With this approach, the space complexity of the visualization process is largely reduced thus allowing visualization to be executed with a reasonable amount of computing resources.

To reduce the space complexity for data visualization, we investigate the characteristics of the matrices to be processed and try to reduce the space requirement based on these characteristics. In a web application, the possible number of intrusion detection measures (web resources accessible by users) can be very large. In our digital library example, the number of web resources accessible is in the range of 2,000 to 3,000. However, the set of web resources accessed during a typical user session is usually much smaller than the total number of measures available on the web application. Hence, the resulting matrices will be sparse and with a majority of the elements (i.e. frequency count of web resources access within the sessions) equal zero. From our experiment with the web portal of a digital library, over 99 percent of the value of x_{ij} of the matrix M equal zero. Based on this characteristic of the matrices, we can leverage on some algorithms for sparse matrix computations [11] to further enhance the data visualization process.

Therefore, one effective way to improve the visualization mechanism is to take advantage of the sparseness of the matrix M. However, this advantage is hard to realize in practice because, according to the factor analysis technique as presented in the previous section, visualization is performed on the normalized data set. It is worth noting that the cluster recognition and inspection is applicable only to the normalized matrix $\widetilde{M}$. Unfortunately, the sparse matrix M will no longer be sparse after normalization. That means the visualization scheme cannot be improved unless the factor analysis algorithm is applied to the sparse matrix M instead of the normalized matrix $\widetilde{M}$.

Referring back to the data visualization algorithm presented in the previous section, the most intensive computation is dedicated to the matrix multiplication $\widetilde{M}^T\widetilde{M}$. As stated, $\widetilde{M}$ is transformed from M with the row vectors normalized. It has been pointed out that M is a sparse matrix, however, $\widetilde{M}$ will not be sparse

after normalization. Thus, we need to investigate the matrix multiplication steps carefully in order to make use of the sparseness of the original matrix M.

Theorem 1. *Let the matrix*

$$M = [x_{ij}], \quad 1 \le i \le n, 1 \le j \le p \tag{6}$$

and

$$\widetilde{M} = [\widetilde{x}_{ij}], \quad 1 \le i \le n, 1 \le j \le p \tag{7}$$

be the row-wise normalized matrix of M. Then

$$\widetilde{M}^T\widetilde{M} = Y^TY - \begin{pmatrix} \varepsilon_1 \cdots \varepsilon_1 \cdots \varepsilon_1 \\ \vdots \quad \vdots \quad \vdots \\ \varepsilon_s \cdots \varepsilon_s \cdots \varepsilon_s \\ \vdots \quad \vdots \quad \vdots \\ \varepsilon_p \cdots \varepsilon_p \cdots \varepsilon_p \end{pmatrix} - \begin{pmatrix} \varepsilon_1 \cdots \varepsilon_t \cdots \varepsilon_p \\ \vdots \quad \vdots \quad \vdots \\ \varepsilon_1 \cdots \varepsilon_t \cdots \varepsilon_p \\ \vdots \quad \vdots \quad \vdots \\ \varepsilon_1 \cdots \varepsilon_t \cdots \varepsilon_p \end{pmatrix} + c \tag{8}$$

where

$$\alpha_k^2 = \sum_{r=1}^{p} (x_{kr} - \overline{x}_k)^2, \quad \beta_k = \overline{x}_k, \quad 1 \le k \le n \tag{9}$$

$$\varepsilon_t = \sum_{k=1}^{n} \frac{\beta_k x_{kt}}{\alpha_k^2}, \quad 1 \le t \le p \tag{10}$$

$$c = \sum_{k=1}^{n} \frac{\beta_k^2}{\alpha_k^2} \tag{11}$$

$$Y = [y_{kt}], \quad y_{kt} = \frac{x_{kt}}{\alpha_k}, \quad 1 \le k \le n, 1 \le t \le p. \tag{12}$$

Proof *Given $\widetilde{M}$ is the row-wise normalized matrix of M, then*

$$\widetilde{M}^T\widetilde{M} = [\widetilde{m}_{st}], \quad 1 \le s, t \le p \tag{13}$$

with

$$\widetilde{m}_{st} = \sum_{k=1}^{n} \widetilde{x}_{ks}\widetilde{x}_{kt} \tag{14}$$

$$= \sum_{k=1}^{n} \frac{(x_{ks} - \overline{(x)}_k)}{\sigma_k\sqrt{p}} frac(x_{kt} - \overline{x}_k)\sigma_k\sqrt{p} \tag{15}$$

$$= \sum_{k=1}^{n} \frac{x_{ks}x_{kt} - \overline{x}_k x_{ks} - \overline{x}_k x_{kt} + \overline{x}_k^2}{\sum_{r=1}^{p} (x_{kr} - \overline{x}_k)^2} \tag{16}$$

With $\alpha_k^2, \beta_k, \varepsilon_t, c, y_{kt}$ *as defined, then*

$$\widetilde{m}_{st} = \sum_{k=1}^{n} \frac{x_{ks}x_{kt}}{\alpha_k \alpha_k} - \varepsilon_s - \varepsilon_t + c \tag{17}$$

$$= \sum_{k=1}^{n} y_{ks}y_{kt} - \varepsilon_s - \varepsilon_t + c \tag{18}$$

□

Theorem 1 allows us to improve the efficiency of the visualization process by substantially reducing the space complexity of the factor analysis algorithm. Due to the results of Theorem 1, the matrix multiplication $\widetilde{M}^T\widetilde{M}$ can be processed as follow:

1. Given matrix M, compute $\beta_k, \beta_k^2, \alpha_k^2, \alpha_k$ for $k = 1 \cdots n$.
2. Compute ε_t for $t = 1 \cdots p$.
3. Compute the constant c.
4. Compute the sparse matrix Y.
5. Compute the matrix multiplication Y^TY.
6. Compute the matrix multiplication $\widetilde{M}^T\widetilde{M}$ according to Equation 8.

Note that the matrix M is sparse and so is Y. To appreciate the improvement due to Equation 8, we analyze and compare the space complexity of the matrix multiplication using both the straightforward method and the new method.

For the memory requirements, if the straightforward multiplication $\widetilde{M}^T\widetilde{M}$ is used, the memory requirement for storing $\widetilde{M}$ is $O(np)$ and for storing $\widetilde{M}^T\widetilde{M}$ is $O(p^2)$. Hence, the total memory requirement will be $O(np+p^2)$. If Equation 8 is used, since Y is sparse and if r is the ratio of non-zero elements to the size of the matrix, then $r \ll 1$. The memory requirement for storing Y is $O(npr)$ and for storing $\beta_k, \beta_k^2, \alpha_k^2, \alpha_k$ and ε_t will be $O(4n+p)$. Totally, the memory requirements will be $O(npr + p^2 + 4n + p)$. Since $r \ll 1$ and $n \gg p \gg 1$, hence, the memory requirements for the matrix multiplication using Equation 8 is much smaller than the straightforward multiplication of $\widetilde{M}^T\widetilde{M}$.

4 Experimental Analysis

The proposed data visualization algorithm was implemented and experiments on its performance were conducted. Audit data from the access log of our university digital library portal were used for the experiments. The library portal has about 100,000 clicks per day. Audit data of one week (from 20 Feb 2003 to 27 Feb 2003) with a total of 665,902 access records were extracted. The corresponding matrix M has 2,742 columns and 9,717 rows. The ratio of non-zero entry, r, of the

matrix M is 0.905%. The experiments were conducted on a high-end PC with dual 1GHz Pentium CPUs and 1G bytes memory. One CPU and 500M bytes is reserved for the execution of the computations.

Figure 3.a depicts the 2-D projection of the points of M using the traditional data visualization algorithm while Figure 3.b depicts the same projection using the new algorithm. The results show that the new algorithm performs as good as the traditional algorithm. That means the new algorithm achieves substantial reduction in space complexity without compromising projection quality.

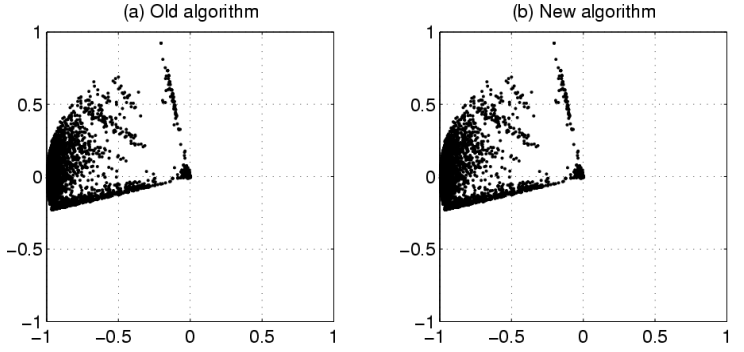

Fig. 3. 2-D projection of points represented by row vectors in matrix M.

To highlight the performance of the new algorithm, two sample data sets were extracted from M. In Data Set 1, the number of measures was fixed while 9 samples with varying numbers of sessions were extracted. In Data Set 2, the number of sessions was fixed while 9 samples with varying numbers of measures were extracted. The memory requirements of the computations are presented in Figure 4. The two data sets are as follow:

Data Set 1				Data Set 2			
Matrix Name	Number of Sessions	Number of Measures	r	Matrix Name	Number of Sessions	Number of Measures	r
M_{11}	1000	2700	1.226%	M_{21}	5000	300	0.503%
M_{12}	2000	2700	1.089%	M_{22}	5000	600	0.424%
M_{13}	3000	2700	0.923%	M_{23}	5000	900	0.495%
M_{14}	4000	2700	0.903%	M_{24}	5000	1200	0.661%
M_{15}	5000	2700	0.894%	M_{25}	5000	1500	0.856%
M_{16}	6000	2700	0.874%	M_{26}	5000	1800	0.915%
M_{17}	7000	2700	0.868%	M_{27}	5000	2100	1.033%
M_{18}	8000	2700	0.928%	M_{28}	5000	2400	0.943%
M_{19}	9000	2700	0.929%	M_{29}	5000	2700	0.894%

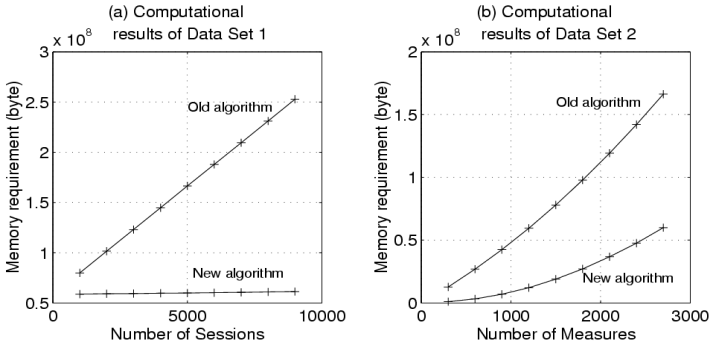

Fig. 4. Comparison of memory requirements for old and new algorithms.

From the results as depicted in Figure 4, it is obvious that the memory requirements were lower for the new algorithm with both varying number of sessions and measures. The improvement is more significant for Data Set 1. It indicates that for the traditional algorithm, the memory consumption grows linearly with the increase of number of sessions. Whereas, with the new algorithm, the increase in memory consumption is much more moderate when number of sessions increases. This is a significant improvement since, as mentioned, visualization of multivariate audit data is an important step in the profiling process. Therefore, the new algorithm out-performs the traditional algorithm significantly and is more suitable for use in practical intrusion detection systems.

5 Conclusion

Multivariate data analysis is an important tool in statistical intrusion detection systems. Intrusion detection typically involves a profiling process, an intrusion detection process and an alert inspection process. Behavior profile of the target system is usually established by statistical analysis techniques or correlation analysis techniques. Intrusion detection alert is raised if the system behavior deviates from the established pattern. Thus creation of the profile and close inspection of its deviation from ongoing activities are of great importance to the performance of the intrusion detection scheme.

Multivariate statistical intrusion detection systems usually require visualization of the multivariate audit data in order to facilitate close inspection by security administrators during profile creation and intrusion alerts. Due to the importance of the profile creation process, security administrators tend to inspect the established profile closely and perform fine-tuning of the profile if possible. Besides, during intrusion detection, the security administrator will also inspect the multivariate audit data closely in order to

decide the event warrants an escalation of the incident to a higher level in accordance with the organization's security policy. Visualization is therefore commonly required as a means for close inspection of the multivariate audit logs.

This paper presented an efficient implementation technique for presenting multivariate audit data needed by statistical-based intrusion detection systems. Though visualization of multivariate data has been in use in the past for supporting statistical-based intrusion detection systems, the need for performing host-based intrusion detection on web-based Internet applications introduced new challenges to the problem. As explained in this paper, the number of intrusion detection measures that need to be analyzed by the multivariate analysis technique increases substantially due to the large number of URL resources managed by typical web servers. Thus, when applying the intrusion detection scheme to web-based Internet applications, the space complexity of the visualization process is usually prohibiting. In order for the approach to be adopted effectively in practice, an efficient technique that allows manipulation and visualization of a large amount of multivariate data is needed. The efficient technique presented in this paper is proven to achieve the same level of visualization quality, and experimental results reported in this paper showed that the new technique greatly reduces the space requirement of the visualization process, thus allowing the approach to be adopted for monitoring web-based Internet applications.

Acknowledgement. We thank the System Division of the Tsinghua University Library for their support in the experiments of this research. This research was partly funded by the National 863 Plan (Projects Numbers: 2001AA414220, 2001AA414020), P. R. China and the Computer Systems Intrusion Detection project of PrivyLink International Limited (Singapore).

References

1. Richard Power. "2002 CSI/FBI Computer Crime and Security Survey". *http://www.gocsi.com*, 2002.
2. D.E. Denning, "An intrusion detection model". *IEEE Trans on Software Engineering*, SE-13, pp 222–232, 1987.
3. R.K. Cunningham, R.P. Lippmann, D.J. Fried, S.L. Garfinkel, T. Graf, K.R. Kendall, S.E. Webster, D. Wyschogrod, M.A. Zissman, "Evaluation Intrusion Detection Systems without Attacking your Friends: The 1998 DAPRA Intrusion Detection Evaluation". Lincoln Laboratory, MIT, USA.
4. E. Biermann, E.Cloete, L.M. Venter. "A Comparison of Intrusion Detection Systems". *Computers & Security*, 20, pp.676–683, 2001.
5. N. Ye, S. M. Emran, Q. Chen, S Vilbert. "Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection". *IEEE Trans. on Computers*, Vol. 51, No. 7, 2002.
6. Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, Mingming Xu. "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data". *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, Vol.31, No.4, 2001.

7. Kwok-Yan Lam, Lucas Hui, Siu-Leung Chung. "Multivariate Data Analysis Software for Enhancing System Security". *J. Systems Software*, Vol.31, pp 267–275, 1995.
8. S. De Backer, A. Naud, P. Scheunders. "Non-linear dimensionality reduction techniques for unsupervised feature extraction". *Pattern Recognition Letters*, 19, pp 711–720, 1998.
9. L. Girardin, D. Brodbeck. "A visual appraoch for monitoring logs". *Proc. of the Twelth Systems Administration Conf.*, p. 299, USENIX Association: Berkeley, CA, 1998.
10. Bill Jacob. *Linear Algebra*. New York: Freeman, 1990.
11. Golub G H, Yon Lean C.F. *Matrix Computation*. John Hopkins Univ Press, 1983.

An IP Traceback Scheme Integrating DPM and PPM

Fan Min, Jun-yan Zhang, and Guo-wie Yang

College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China
{minfan, xindyzhang, gwyang}@uestc.edu.cn

Abstract. IP traceback technology is an important means combating Denial of Service (DoS) attacks in Internet. This paper proposes a new IP traceback scheme constituting two parts: the first part is constructing a traceback tree by integrating Deterministic Packet Marking and Probabilistic Packet Marking, and the second part is getting attack routes by analyzing this traceback tree. Basing on performance analysis, we point out that our scheme is both efficient and robust against mark field spoofing.

Keywords: Denial of Service, IP spoofing, IP traceback, Deterministic Packet Marking, Probabilistic Packet Marking

1 Introduction

A "Denial-of-Service" (DoS) attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service [1]. The attackers may use one or more of these ways to achieve their aims: a, bring down the machine providing the service; b, flood the link between the user and the service provider; c, utilize resources available at server using bogus requests [2]. Distributed Denial of service (DDoS) attacks constitutes one of the single greatest threats facing businesses involved in electronic commerce [3].

DoS attackers usually hide their actual address by IP forging, especially IP source address spoofing. Ingress / Egress filtering is effective method against IP forging. In ingress filtering, a router does not allow a packet into the network, if it has a source IP address of a node inside the network. On enabling egress filtering, a router sends packets to Internet only if the source address of the packet is that of a node within that network [4]. If ingress filtering is installed, spoofed IP address has to be in the same network part in order to pass through the router. Egress filtering is a complement of ingress filtering. The principal problem with ingress filtering is that its effectiveness depends on widespread, if not universal, deployment. A secondary problem is that even if ingress filtering were universally deployed at the customer-to-ISP level, attackers could still forge addresses from the hundreds or thousands of hosts within a valid customer network [5].

IP traceback is an important method finding actual addresses of IP forging attackers, also called anonymous attackers. If attack source could be found, attacking packet from corresponding computer can be abandoned, and attack could be prevented, legal issues could also be involved for corresponding person.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 76–85, 2003.

2 Relative Works

Packet marking [5-8] is a new IP tracing back scheme in recent years, it can be further classified into Deterministic Packet Marking (DPM) and Probabilistic Packet Marking (PPM).

2.1 DPM

Using IP Record Router option [10] in IP packet, each router in the network will write its own IP address into option field of the IP packet. And destination host can get full route of correspond packet by simply investigating this field. This scheme is called DPM.

Shortcomings of DPM include: a, increases packet length dramatically. For about 20 hops the packet size will increase by 80 bytes (IPv4). An overhead of 80 bytes for every packet (about 500 bytes) is unacceptable [6]; b, since the length of the path is not known a priori, it is impossible to ensure that there is sufficient unused space in the packet for the complete list [5]; c, increases router operation.

2.2 PPM

PPM [6-8] is a scheme trying to make up shortcomings of DPM, it is supposed to deal with type b and c attack in Section 1. It is further classified into:

1. *Node Sampling*. A single static "node" field is reserved in the packet header—large enough to hold a single router address (i.e., 32 bits for IPv4). Upon receiving a packet, each router chooses to write its address in the node field with some probability *p* [5].
2. *Edge Sampling*. Reserve two static address-sized fields, *start* and *end*, in each packet to represent the routers at each end of a link, as well as an additional small field to represent the distance of an edge sample from the victim [5]. Upon receiving a packet, each router chooses to write its address in the *start* field with some probability *p*, and the *end* field is written by the next router if *start* field is written by the previous router. Distance from the edge to destination can be easily gotten through adding up [8].

In IPv4, 72 bits is needed for two address-sized fields and a distance field. Savage [5] uses coding method to compress it into 16 bits and put into *ID* field of IP header. Complexity is incurred by coding. In the following context, PPM all refer to node sampling.

Advantages of PPM over DPM includes: a, only one address-size field is needed; b, packet size does not change on the flight; c, a router only mark a packet with probability *p*, incurring less operation.

Disadvantages of PPM includes: a, getting a full path from victim to attacker is a rather slow progress; b, for DDoS attacks, this method may fail [5].

3 Integrated Marking and Tracing Scheme

In this section we present an integrated marking and tracing scheme based on DPM and PPM.

3.1 Assumptions

Let A denote attacking host, A is not unique; let V denote victim, V is unique; let T1 and T2 denote two types of ICMP packets, they all have *IP Record Router option* set.

Our scheme relies on the following assumptions. In most cases, these assumptions hold true.

[Assumption 1] Source address and marking field of attacking packets are spoofed.

Source address spoofing is the fundamental reason of IP traceback. Attackers usually choose spoofing IP source address except DDoS with very good distribution. Marking field spoofing can effectively impede traceback by the victim [8].

[Assumption 2] Routing does not change in a short period.

Theoretically network topology can change at any time. But changes of important network elements such as routers do not happen frequently. In short period (e.g. a few minutes), the probability of routing change is very small.

[Assumption 3] Let *NS* denote maximal packet sending speed of ordinary host to V; let *P* denote packet sending speed of any single A to V; let *K* denote attack volume factor; and $P > NS \cdot K$.

In order to make attack effective, for situation that number of A is not large, attackers may set $K > 100$.

[Assumption 4] V is unable to distinguish whether or not a packet is attack packet simply by data in the packet.

3.2 Scheme Description

Our scheme constitutes three closely related algorithms.

Any router runs Algorithm 1 upon receiving a packet:

1. If current packet is ordinary packet, and the destination address is not this router, fill IP address of this router into *node* field of this packet with probability *p*, and forward it; /*The same as PPM*/
2. If this packet is T1 packet, and the destination is not this router, write IP address into option field of this packet and forward it; /*The same as DPM*/
3. If this packet is T1 packet, and the destination is this router,

3.1 If the *Key* in the *Key* field is not generated by this router recently, exchange source and destination addresses of this packet, write IP address into option field of this packet and send it out;

3.2 If the *Key* in the *Key* field is generated by this router recently, store it for further use.

Algorithm 1. Marking algorithm run by routers.

When V discovers that packet arrival rate excesses a given bound, it concludes that it is under attack, and runs Algorithm 2.

1. Stop service;
2. Generate a key *Key*;
3. In time Δt, for each arrival packet, generate a T1 packet, copy the mark field of the arrival packet into destination address of the ICMP packet, copy *Key* into corresponding field, send it out;
4. Run Algorithm 3, analyze arrival T1 packets containing *Key*;
5. According to the analysis result, send T2 packets to routers nearest to A, and these routers will abandon packets whose destination is V.

Algorithm 2. Trace algorithm run by V

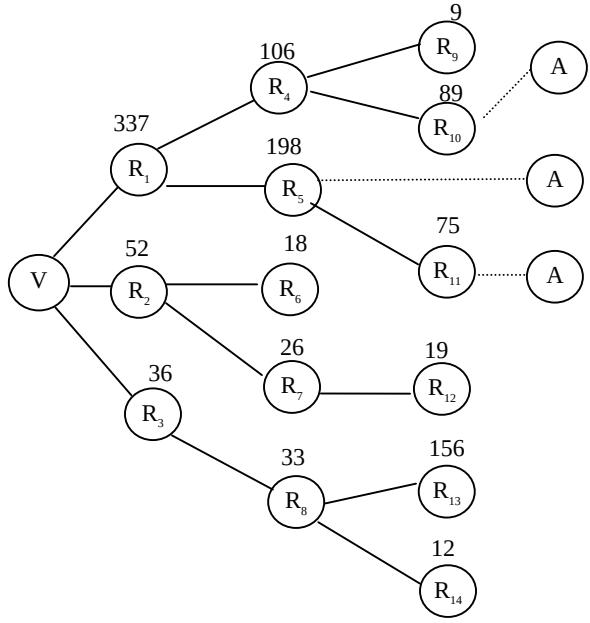

Fig. 1. An example of network topology

Upon receiving T1 packets from routers, V gets a series paths from which a tree taking V as root is constructed. Fig. 1 is an example of such tree, A is not included.

In Fig. 1, data above each node (called marking number) represents T1 packets received by V with corresponding node as source in Δt. Next we describe T1 packet analysis algorithm using Fig. 1.

1. According to node level with ascending order: let *MP* denote marking number of current node, let *C* denote its children number; let MC_i denote marking number of

children subject to $MC_1 \geq MC_2 \geq ... \geq MC_C$. If $MP < \sum_{i=1}^{C} MC_i$, decrease MC_1, MC_2, ..., MC_C until $MP = \sum_{i=1}^{C} MC_i$. In fig. 1, marking number of R_{13} should be changed to 21. For another example, suppose node N's marking number is 56, marking numbers of its four children is 132, 77, 15 and 9, these numbers should be changed to 16, 16, 15 and 9 respectively.

2. According to node level with descending order: if marking number of a node is smaller than $1/\alpha$ of that of its brother, delete sub-tree rooted at this node. We call α *safe speed factor 1*, and it's reasonable to let $\alpha \in [5, 10]$. If $\alpha = 5$, in fig. 1 R_9 should be deleted;
3. According to node level with descending order: if a node is already a leaf, and its marking number is less than $1/\beta$ of average marking number of the same level. We call β *safe speed factor 2*, it's reasonable to let $\beta \in [2, 3]$. If $\beta = 2$, in Fig. 1 R_{12}, R_{13}, R_{14}, R_6, R_7, R_8, R_2, R_3 should be deleted with corresponding order;
4. The remaining leaves are adjacent with A. In Fig. 1 they are R_{10} and R_{11};
5. According to node level with decreasing order: if a node has marking number greater than double of its children marking number sum, this node is also adjacent with A. In Fig. 1 it is R_5.

Algorithm 3. T1 packet analysis algorithm run by V.

3.3 Scheme Explanation

In essence, this scheme is an integration of PPM and DPM, in ordinary cases PPM is occupied, while routers use DPM on demand of V after DoS attack is detected.

Next we explain Algorithm 3 in detail. Let N denote node in context; because node has a unique address, let *N* denote address of N; let *d* denote hop count from N to V.

1. According to assumption 4, only packet sending speed from N to V can be collected to judge if N is adjacent with A;
2. From Algorithm 1 we can see that for IP packet from the same source, upon receiving it in V, the probability of the marking field be N is $p \cdot (1-p)^{d-1}$, be a child of N is $p \cdot (1-p)^{d}$. So we have

$$MP \cdot (1-p) = \sum_{i=1}^{C} MC_i \tag{1}$$

Obviously $0 < p < 1$, we have

$$MP \geq \sum_{i=1}^{C} MC_i \tag{2}$$

So only two kinds of reasons may lead to $\sum_{i=1}^{C} MC_i > MP$: a, because of uncertainty of probability. In this situation, a little modification of a few relatively large marking numbers does not influence result of the scheme; b, A forged a large amount of packets using some children of N as source address, and the most suspicious ones are those who have large marking numbers. So step 1 can greatly eliminate marking field spoofing.

3. Step 2, 3 are based on Assumption 4;
4. A router can receive attack packet directly or indirectly, which is the very basis of step 5.

3.4 Performance Analysis

Let T_L denote time from detecting attack to locate routers nearest to A. It constitutes three parts: I, time for collecting enough attack packet, Δt; II, time from sending T1 packets to receiving them; III, time needed running Algorithm 3. Next we analyze these three parts one by one:

I. Let A_i denote a specific attacker. To confirm that N is nearest to A_i, V needs a number of packets with marking field N. Let R_N denote packets needed for analysis, R_A denote packet from A_i, we have

$$R_N = R_A \cdot p\,(1-p)^{d-1} \tag{3}$$

Let packet generating speed of A_i be P, we have

$$\Delta t = \frac{R_A}{P} = \frac{R_N}{P \cdot p\,(1-p)^{d-1}} \tag{4}$$

R_N is related with accuracy of allocating A_i, R_N is determined by V. Routers can set p to minimize Δt. Let

$$f(p) = p\,(1-p)^{d-1} \tag{5}$$

$$f'(p) = (1-p-d\cdot p+p)(1-p)^{d-2} = (1-d\cdot p)(1-p)^{d-2} \tag{6}$$

Clearly when

$$p = \frac{1}{d} \tag{7}$$

$f(p)$ is maximized, and Δt is minimized.

In Internet environment, $d = 20$ is a widely used setting [8]. Under this condition we should set $p = 0.05$, if we further set $R_N = 50$, then

$$\Delta t = \frac{2650}{P} \tag{8}$$

Specifically, Δt = 5.3 (seconds) while P = 500 (packets / second); Δt = 88.3 (seconds) while P = 30 (packets / second).

If V can cache recently received packets, Δt can decrease accordingly.

II. The second part is related with channel speed. In most cases, it should be less than 1 second. Moreover, this time is overlapped with the first part.

III. Algorithm 4 only involves single loop, the time complexity is

$$O(ND) \tag{9}$$

ND is the number of routers sending back T1 packet to V. Run time of Algorithm 4 should also be less than 1 second.

In summary, T_L is mainly decided by Δt, while the latter is mainly decided by P. When number of A is relatively small, to make attack more effective, P is relatively large, and T_L can be under 10 seconds; when number of A is relatively large (well distributed DDoS), P is relatively small, and T_L may be tens of seconds or more. Another important problem incurred by small P is that K can also be small (e.g., less than 10), and our scheme can not distinguish between attack packets and ordinary packets.

3.5 Influence of Spoofed Marking Field

In PPM, the probability of receiving a marked packet from the farthest router is $f(p)$ (see Equation (5)), and the probability of receiving a packet never marked (potentially being spoofed by the attacker) is

$$g(p) = (1 - p)^d \tag{10}$$

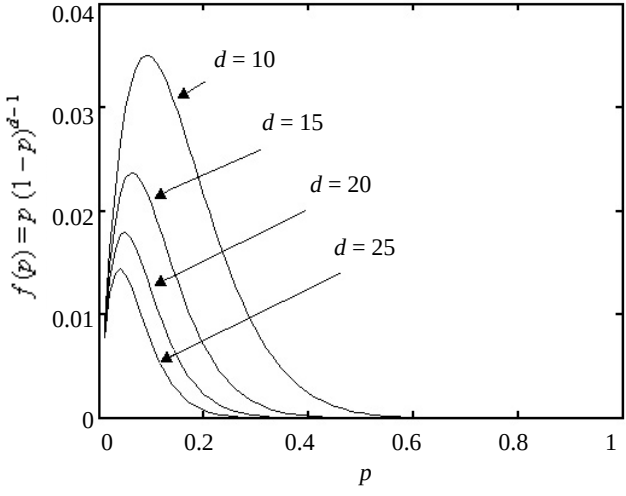

Fig. 2. Probability of receiving packets marked by the farthest router

For effectiveness, we want to maximize $f(p)$, which in turn requires $p=1/d$ (see Equation (7) and Fig. 2). While for robustness, we want to minimize $g(p)$, which in turn requires $p = 1$ (See Fig. 3).

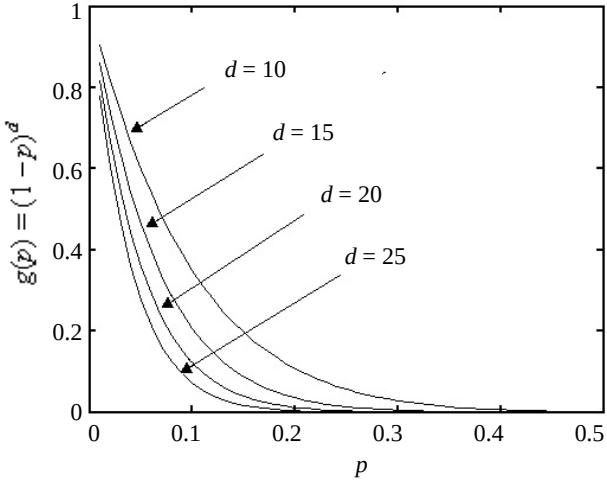

Fig. 3. Probability of receiving packets never marked

For example, if $d = 20$, and let $p = 1/d$, then $f(p) = 0.01887$, which is maximized; but $g(p) = 0.3585$, which means that a large proportion of attack packets would arrive at victim with forged mark address unchanged, and effectiveness of PPM would be significantly threatened.

In order that mark field spoofing has no essential influence, we should set $g(p) > f(p)$, which in turn requires $p > 0.5$. But this is inefficient. For example, let $d = 20$, $p = 0.51$, then $f(p) = 6.627\times10^{-7}$, $g(p) = 6.336 \times10^{-7}$, and $1 / f(p) \approx 1.5\times10^{6}$ packets are needed to get a marked packet from the furthest router.

Our scheme uses T1 packets instead of PPM packets for path reconstruction. In order to interference our scheme, attacker must deliberately spoof marking field as follows: a, use router IP (or else no corresponding T1 packet would return to V); b, router addresses or spoofed packet should comply with equation (2). These two conditions require that attackers have good knowledge about networks topology around V, which is very hard to achieve.

Accordingly, robustness of our scheme is not influenced by $g(p)$.We can set $p = 1/d$ to maximize $f(p)$, and only $1 / f(p) = 1 / 0.01887 \approx 53$ packets are needed to get a full path from the furthest router when $d = 20$.

3.6 Advantages of Our Scheme

Main advantages of our scheme include:

1. Effectiveness against DDoS with relatively large K;
2. Rapid reaction. In most cases less than tens of seconds;

3. Still valid after attack stops. Only Δt time is needed to collect enough attack packets;
4. More economy than DPM. Use DPM only for T1 packets under attack;
5. More robust than PPM, and can effectively eliminate influence of marking field spoofing;
6. No need of human interfering. All operation can be done by routers and V automatically;
7. T1 packet has little data. Added up router addresses would not excess IP packet length bound;
8. Because of *Key* (though the use of key is very simple), it's hard for attacker to spoof T1 packets. Besides, spoof T1 packet (or T2 packet) is a dangerous operation for attackers because full path is recorded.

4 Discussions

On constructing a traceback tree, V may send out a large number of T1 packets, many of which having identical destination addresses. From Assumption 2 and Algorithm 1 we can see that if the initial T1 packets are identical, the returned T1 packets are also identical. For example, in Fig. 1 V received 89 T1 packets from R_{10}, all recording the whole path from V to R_{10} and R_{10} to V. This is a waste of network resources. For example, in Fig. 1 a total of 1166 T1 packets are transferred.

We can revise Algorithm 2 (step 3) to send out only a few (more than one to ensure that one of them can fulfill the task) T1 packets per marking address, and at the same time calculate number of packets having the same marking address, in this way the traceback tree can also be successfully constructed, and bandwidth and router operation is largely saved. For example, in Fig. 1 if V sends 2 T1 packets per marking address, only $2 \times 14 = 28$ T1 packets should be transferred.

T2 packets are vital for networks behavior. In addition to DMP, stronger security mechanisms such as authentication are needed for secure transferring of T2 packets.

5 Conclusions

In this paper, we integrate DPM and PPM and get a new IP traceback scheme. Performance of our scheme is analyzed. Our scheme has advantages such as rapid reaction, robustness, and can efficiently combat DoS attacks. It can get even better performance if Assumption 4 does not hold true.

References

1. "Denial of Service Attacks", CERT Coordination Center, Oct 1997. Available at http://www.cert.org/tech_tips/ denial_of_service. html

2. K. J. Houle, G. M. Weaver, N. Long, R. Thomas, "Trends in Denial of Service Attack Technology", CERT Coordination Center, October 2001.
3. L. Garber, “Denial-of-service attacks rip the Internet,” Computer, pp. 12–17, Apr. 2000.
4. P. Ferguson and D. Senie, “Network ingress filtering: Defeating denial-of-service attacks which employ IP source address spoofing”, RFC 2827, 2000.
4. "Denial Of Service Attacks – A Survey", CERT Coordination Center, Draft. Work in progress.
5. S. Savage, D. Wetherall, A. Karlin and T. Anderson, “Practical Network Support for IP Trace-back,” Proc. 2000 ACM SIGCOMM, vol. 30, no. 4, ACM Press, New York, Aug. 2000, pp. 295–306.
6. H. Burch and B. Cheswick, “Tracing anonymous packets to their approximate source,” in Proc. 2000 USENIX LISA Conf., Dec. 2000, pp. 319–327.
7. T. Baba and S. Matsuda, “Tracing Network Attacks to Their Sources” IEEE Internet Computing March · April 2002, pp. 20–26.
8. K. Park and H. Lee, “On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack” Proc. IEEE INFOCOM '01, 2001.
9. S. Bellovin, M. Leech, and T. Taylor, "ICMP Traceback Messages", Internet draft, work in progress, Jan. 2003; available online at http://www.ietf.org/internet-drafts/draft-ietf-itrace-03.txt (expires July 2003).
10. J. Postel, “Internet protocol,”, RFC 791, 1981.
11. Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, et. al., “Hash-Based IP Traceback”, Proc. 2000 ACM SIGCOMM
12. D. Dean, M. Franklin, and A. Stubblefield, “An algebraic approach to IP traceback”, Network and Distributed System Security Symposium, NDSS '01
13. Minho Sung, Jun Xu , “IP Traceback-Based Intelligent Packet Filtering: A Novel Technique for Defending against Internet DDoS Attacks”, 10th IEEE International Conference on Network Protocols (ICNP'02) November 12–15, 2002

Improved Scalable Hash Chain Traversal

Sung-Ryul Kim*

Division of Internet & Media
and Multidisciplinary Aerospace System Design Center
Konkuk University
Seoul, Korea

Abstract. Yaron Sella recently proposed a scalable version of Jakobsson's algorithm to traverse a hash chain of size n. Given the hash chain and a computation limit m ($k = m + 1$ and $b = \sqrt[k]{n}$), Sella's algorithm traverses the hash chain using a total of kb memory. We improve the memory usage to $k(b-1)$. Because efficient hash chain traversal algorithms are aimed at devices with severely restricted computation and memory requirements, a reduction by a factor of $(b-1)/b$ is considered to be important. Further, our algorithm matches the memory requirements of Jakobsson's algorithm while still remaining scalable. Sella's algorithm, when scaled to the case of Jakobsson's algorithm, has a memory requirement of about twice that of Jakobsson's.

Keywords: efficient hash chain traversal, secure hash, pebbles

1 Introduction

Many useful cryptographic protocols are designed based on the *hash chain* concept. Given a *hash function* $f()$ which is assumed to be difficult to invert, a hash chain is a sequence $< x_0, x_1, \ldots, x_n >$ of values where each value x_i is defined to be $f(x_{i-1})$. The hash chain is being used as an efficient authentication tool in applications such as the S/Key [2], in signing multicast streams [5], message authentication codes [5,6], among others.

Traditionally the hash chain has been used by one of two methods. One is to store the entire hash chain in memory. The other is to recompute the entire hash chain from x_0 as values are exposed from x_n to x_0. Both methods are not very efficient. The first one requires a memory of size $\Theta(n)$ while the second one requires $\Theta(n)$ hash function evaluations for each value that is exposed. The memory-times-storage complexity of the first method is $O(n)$ and it is $O(n^2)$ for the second method. As mobile computing becomes popular, small devices with restricted memory and computation powers are being used regularly. As these devices are being used for jobs requiring security and authentication, the memory and computation efficiency of hash chain traversal is becoming more important.

* Corresponding author. kimsr@konkuk.ac.kr

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 86–95, 2003.

Influenced by amortization techniques proposed by Itkis and Reyzin [3], Jakobsson [4] proposed a novel technique that dramatically reduces the memory-times-storage complexity of hash chain traversal to $O(\log^2 n)$. The algorithm by Jakobsson uses $\lceil \log n \rceil + 1$ memory and makes $\lceil \log n \rceil$ hash function evaluations for each value that is exposed. The result has been further improved by Coppersmith and Jakobsson [1], to reduce the computation to about half of the algorithm in [4]. Both results are not intended for scalability. That is, the amount of computation and memory depends only on the length of the chain n and they cannot be controlled as such needs arise. For example, some device may have abundant memory but it may be operating in a situation where the delay between exposed hash values are critical. To address the issue Sella [7] introduced a scalable technique that makes possible a trade-off between memory and computation requirements. Using the algorithm in [7], one can set the amount m of computation per hash value from 1 to $\log n - 1$. The amount of memory required is $k\sqrt[k]{n}$ where $k = m + 1$. In the same paper, an algorithm designed specifically for the case $m = 1$ is also presented. The specific algorithm reduces the memory requirement by about half of that in the scalable algorithm.

We improve the memory requirement of Sella's scalable algorithm. While Sella's algorithm has the advantage of scalability from $m = 1$ to $\log n - 1$, the memory requirement is about twice that of Jakobsson's algorithm when $m = \log n - 1$. Our algorithm reduces the memory requirement of Sella's algorithm from $k\sqrt[k]{n}$ to $k(\sqrt[k]{n} - 1)$. This might not seem to be much at first. However, if k is close to $\log n$ the reduction translates into noticeable difference. For example, if $k = \frac{1}{2} \log n$, the memory requirement is reduced by a factor of $\frac{1}{4}$. If $k = \log n$, the memory requirement is reduced by half. Of particular theoretical interest is that our algorithm exactly matches the memory and computation requirements of Jakobsson's algorithm if we set $k = \log n$. Our technique has one drawback as the scalability is reduced a little bit. It is assumed that $m \geq \sqrt[k]{n}$, that is, our algorithm does not scale for the cases where m is particularly small.

We note here that, as it is with previous works, we count only the evaluation of hash function into the computational complexity of our algorithm. This is because the hash function evaluation is usually the most expensive operation in related applications. We also note that the computation and memory bounds we achieve (and in the previous works) are worst-case bounds (i.e., they hold every time a value in the hash chain is exposed).

2 Preliminaries

2.1 Definitions and Terminology

We mostly follow the same terminology and basic definitions that appear in [7]. A hash chain is a sequence of values $< x_0, x_1, \ldots, x_n >$. The value x_0 is chosen at random and all other values are derived from x_0 in the following way: $x_i = f(x_{i-1})$, $(1 \leq i \leq n)$, where $f()$ is a hash function. A single value x_i is referred to as a *link* in the chain. The chain is generated from x_0 to x_n but is exposed from x_n to x_0. We use the following directions terminology: x_0 is placed at the

left end, and it is called the *leftmost* link. The chain is generated *rightward*, until we reach the *rightmost* link x_n. The links of the chain are exposed *leftward* from x_n to x_0. The time between consecutive links are exposed is called an iteration. The above hash chain has $n + 1$ links. However we assume that at the outset, x_n is published as the public key. Hence, we refer to $X =< x_0, x_1, \ldots, x_{n-1} >$ as the full hash chain and ignore x_n.

The goal of the hash chain traversal is to expose the links of the hash chain one-by-one from right to left. The efficiency of a traversal is measured in both the amount of memory required and the number of hash function evaluations between each exposed link.

In the work by Sella the concept of a *b-partition* of the hash chain is used. It can be also be used to describe Jakobsson's algorithm. We repeat the definitions here.

Definition 1. *A* b-partition *of a hash chain section means dividing it into* b *sub-sections of equal size, and storing the leftmost link of each sub-section.*

Definition 2. *A* recursive b-partition *of a hash chain section means applying* b*-partition recursively, starting with the entire section, and continuing with the* rightmost *sub-section, until the recursion encounters a sub-section of size* 1.

The b-partition and recursive b-partition are easier to describe when we map the partition to a b-ary tree. We will use the basic terminology pertaining to trees without definition. The following is the definition of the mapping between the recursive b-partition and a b-ary tree. It is repeated also from [7].

Definition 3. *A* recursive b-tree *is a recursive mapping of a recursive* b*-partition onto a tree as follows.*

- *Each node in the tree is a section or a sub-section. In particular, the entire hash chain is represented by the root of the tree.*
- *For every section* L *that is partitioned from left to right by sub-sections* $L_1, L_2, \ldots, L_b$*, the corresponding node* N *is the parent of the corresponding nodes* $N_1, N_2, \ldots, N_b$*, and the children keeping the same left-to-right order.*

See Figure 1 for an example. It shows a recursive 4-tree with 3 levels. The root of the tree corresponds to the entire hash chain and the tree leaves correspond to single links in the hash chain. Even though each node is shown to be storing b links, only $b - 1$ links will be the memory requirement for a node because the leftmost link is stored in the parent.

Sub-sections induced by recursive b-partition are *neighbors* if their corresponding nodes are at the same level and appear consecutively at the natural depiction of the recursive b-tree. Note that they may not share the same parent. Depending on its position in the recursive b-partition, a sub-section can have a *left neighbor*, a *right neighbor*, or both. A node in a recursive b-tree will sometimes be called a sub-section, meaning the corresponding sub-section in the recursive b-partition.

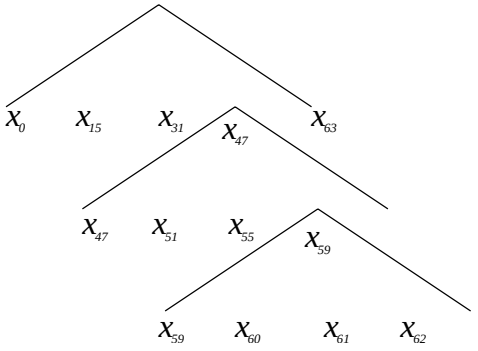

Fig. 1. Example 4-tree with 3 levels

In order to traverse the hash chain we adopt the notion of pebbles and stored links from previous works. A *pebble* is a dynamic link that moves from left-to-right in a hash chain. The mission of a pebble is to induce a b-partition of a sub-section S. That is, it starts at a stored link and traverses the chain while storing the values corresponding to the b-partition of S. The initial stored link will be received from the (sub-)section that S is a part of. For a pebble that induces a b-partition $(x_{i_1}, x_{i_2}, \ldots, x_{i_b})$, the link x_{i_b} is called the *destination* because the pebble does not have to move any further after reaching x_{i_b}.

2.2 Jakobsson's Algorithm

Jakobsson's algorithm can be considered to be using the recursive 2-tree of the entire hash chain. There are $\lceil logn \rceil + 1$ levels in the tree. One pebble is present at each level except at the root. When the algorithm starts, a pebble is placed at the mid-point of the sub-section S at each level.

When the mid-point is exposed the pebble is moved to the start of the left neighbor. From there it traverses the hash chain two links per exposed link. Because it (and all other pebbles) moves at twice the speed that the links are exposed, a pebble reaches the end of the left neighbor when all the links in S has been exposed. It can be shown that at most one of the pebbles in two adjacent levels are moving at any point. This leads to the above mentioned computation cost per iteration. We note that because the pebble starts to move only after its destination is reached and because there are only one link to store (the leftmost node is stored in the parent), the pebble never stores a link and moves to the right. Thus the memory requirement is the same as the number of pebbles (plus one at the root).

2.3 Sella's Algorithm

Sella's algorithm uses a recursive b-partition. As in the Jakobsson's algorithm, one pebble is placed at each level. However, the pebble is moved to the start of the left neighbor immediately when the rightmost link (not the rightmost stored link) in a section is exposed. The pebble moves at the same speed as the links are exposed.

The parameter b is set to be $\sqrt[k]{n}$ where $k = m+1$. That is, the tree will have k levels. Because no pebble is needed at the root, the computational requirement per iteration is m. Initially, there are $b-1$ stored links at each level. However, because pebbles start to move before any stored link is exposed, one memory cell is required for each pebble. Thus, the total memory requirement amounts to $k\sqrt[k]{n}$. If we set $k = \log n$, the memory requirement becomes $2\log n$, which is about twice that of Jakobsson's algorithm.

2.4 Intuitions for Our Algorithm

In our proposed algorithm, each pebble starts to move after the rightmost stored value is used, thus removing the memory requirement for the pebbles. Because it started late (as in Jakobsson's algorithm) it makes accelerated moves. That is, it traverses b links (called *making a move*) while one link is exposed. It cannot be moving all the time because if so, the computational requirement per iteration will be $b \times m$. It makes a move about every $b/(b-1)$ times, enough to catch up to its late start but not at a regular interval as described later.

It is quite simple in Jakobsson's algorithm because there can be no overlap in computation for pebbles residing in two adjacent levels (in the b-tree). We have to consider b adjacent levels. There will be overlaps if we allocate the moves for different levels at regular intervals. So we have to carefully allocate the iterations where each pebble makes a move. The tricky part is to show that there always exists a way to allocate iterations such that a pebble is never too late to its destination.

3 Improved Algorithm

Assume that we have a hash chain $< x_0, x_1, \ldots, x_{n-1} >$ to traverse. Let k be $m+1$ and let b be $\sqrt[k]{n}$. We assume that $k-1$ is a multiple of m and we also assume that $m \geq b$.

3.1 Algorithm

We first form a recursive b-partition (also the corresponding recursive b-tree) off-line. We put a pebble at the rightmost stored link in each sub-section. Then, the following loop is repeated for n times.

- Exposure loop
  1. Expose and discard L, the current rightmost link in the hash chain.
  2. If $(L = x_0)$ stop.
  3. For each pebble p do
     - If L is a rightmost stored link in a sub-section S' and p is on L, then move p to the left end of S where S is the left neighbor of S'.
  4. Call **Schedule Iterations** for all pebbles p starting from the bottom-most level to the top-most level.
  5. Advance all pebbles that are allocated the current iteration by b links.
  6. Repeat from Step 1.

Now we describe the procedure **Schedule Iterations**. We define *level numbers* in the recursive b-tree. The lowest level is said to be at level 1 and the level number increases as we go up the tree. Thus, the root will be at level k. Assume that p is at level t. Let S be the sub-section where p is newly assigned. Let S' be the right neighbor of S. There are b^{t+1} links in S. We give a natural correspondence between a link in S and an iteration where a link in S' is exposed. The leftmost link in S corresponds to the iteration where the rightmost link in S' is exposed. And the correspondence proceeds towards the right in S and towards the left in S'. That is, the links in S and the links in S' are matched in reverse order. When we say that we allocate a link x_i in S to p it means that p will advance b times rightward at the time when the link in S' that corresponds to x_i is exposed. Note that the iterations corresponding to the b^t leftmost links in S have already passed. Let $(L_1, L_2, \ldots, L_b)$ be the b-partition of S.

Our strategy will satisfy the following three conditions. We will shortly verify that the conditions are satisfied by our algorithm.

1. A link should not be assigned to p if it is assigned to a pebble in lower $b-1$ levels. This ensures that at any iteration, at most one pebble in consecutive b levels is making a move. There are many sub-sections in the lower $b-1$ levels but the schedule for each pebble will be the same regardless of the particular sub-section.
2. Pebble p should not move too fast. Because p has to store a link when it reaches a position of a stored link for S, one stored link in S' has to be exposed before we store one link in S.
3. The pebble q at level $t-1$ must have enough links to be allocated after all computation for p is finished. Pebble q cannot be moved until it receives the rightmost stored link in S'. Thus, the schedule for p must finish before the schedule for q starts.

Procedure **Schedule Iterations** works as follows. First, p is not scheduled if any of the pebbles in the lower $b-1$ levels are allocated the current iteration. Second, p is not scheduled if p has to store a link at the current location and there is not enough memory in the current level for the stored link. Third, if p is at the leftmost position of sub-section S and the link at the current position is not available yet. If all three conditions are false and p is yet to reach its destination, then p is scheduled the current iteration. See Fig. 2 for an example.

Fig. 2. Example run of the algorithm for $n = 27$, $k = 3$, $b = 3$.

3.2 Correctness

We show by a series of lemmas that there always exists a schedule that satisfies the conditions mentioned in the algorithm. Let S be the sub-section at level t where p is newly assigned. Also let $(L_1, L_2, \ldots, L_b)$ be the b-partition of S. The following lemma shows that we have enough unallocated links to allocate for pebble p at level t and the $b-1$ pebbles in the lower $b-1$ levels (condition 1).

Lemma 1. *There are enough links in S for the b pebbles p and in lower $b-1$ levels.*

Proof. By the time p is moved to the start of S the iterations for the links in L_1 have already passed. So there are a total of $b^t(b-1)$ links to be allocated. p has to move across $b^t(b-1) = b^{t+1} - b^t$ links to reach its destination. Thus, it has to make $b^t - b^{t-1}$ moves (same number of links are to be allocated). At level $t-1$, there are $b-1$ sub-sections to schedule and in each sub-section there are $b^{t-1} - b^{t-2}$ moves to be made by the pebble in level $t-1$. So at level $t-1$ there are $b^t - 2b^{t-1} + b^{t-1}$ moves to be made in total. From then on to the level $t-b+1$, the same number of moves are to be made.

By adding them up, we can see that we need $b^{t+1} - 2b^t + 2b^{t-1} - b^{t-2}$ links to allocate to the pebbles. Subtracting this number from the number of available links $b^{t+1} - b^t$ we have the difference $b^t - 2b^{t-1} + b^{t-2}$, which is always positive if $b \geq 2$.

The following lemma and corollary shows that the memory requirement (condition 2) is satisfied.

Lemma 2. *There are enough links in L_b for the last b^{t-1} moves of p.*

Proof. There are b^t links that can be allocated in L_b. Pebble p at level t requires b^{t-1} moves by the condition of the lemma. The pebble at level $t-1$ has to make $b^{t-1} - b^{t-2}$ total moves. From then onto the next $b-2$ levels, the same number of moves are needed. Adding them up leads to $b^t - b^{t-1} + b^{t-2}$. Substracting this number from the available number of links b^t results in

$$b^{t-1} - b^{t-2}, \tag{1}$$

which is positive if $b \geq 2$.

Corollary 1. *Pebble p can be moved slow enough so that the memory requirement for level t is $b-1$.*

Proof. The above lemma shows that the last b^t links to be traversed by p can be actually traversed after all the stored links in the right neighbor of S. Even if we assume that the first $b^t(b-2)$ moves are allocated as far to the right as possible, there are enough number of links to allocate for p, and so p will move at a regular pace at the worst case. Thus, p can be moved slow enough to satisfy the memory requirement.

Even if we have shown that there are enough number of links to allocate for every pebble, it is not enough until we can show that condition 3 is satisfied. Consider a pebble r at level 1, because no pebble needs to be scheduled after r, r can be allocated the rightmost link in the sub-section. A pebble r' at level 2 has to finish its computation before r has to start its computation. Thus, there is a leftmost link (called the *leftmost allocatable link*) for the pebble in each level where the pebble has to start its computation to reach its goal early enough for the pebble in the lower level. Conversely, we have to make sure that the leftmost link allocated to a pebble at every level is to the right enough so that the upper levels have enough room to finish its moves. The following lemma shows that there are enough links to the left of the leftmost link allocated for pebble q at level $t-1$ (condition 3). Note that the final difference in the proof of the above lemma will be used in the proof of the next lemma.

Lemma 3. *There is enough links to allocate to p and the $b-1$ pebbles in the lower levels to satisfy condition 3.*

Proof. As mentioned above, the leftmost allocatable link for level 1 is the first from the rightmost link in a sub-section. For level 2, it is the b-th link from the rightmost link. If $b = 2$, the leftmost allocated link for level 3 ls b^2-th one from the rightmost link. The lemma is easy to prove if $b = 2$. In that case, the leftmost allocatable link for level m is b^{m-1}-th link from the rightmost link in a sub-section. However, if $b > 2$, the leftmost allocated link for level 3 is $(b^2 + 2b)$-th one from the rightmost link because $2b$ links that are already allocated to the pebble at level 1 cannot be allocated. We can prove by induction that the leftmost link for level m is at most the $(b^{m-1} + 2b^{m-2} + 2^2 b^{m-3} + \cdots)$-th one from the rightmost link in a sub-section. This number sums to at most $b^m/(b-2) < b^m$.

In the proof of lemma 2, we have $b^{t-1} - b^{t-2}$ links that need not be allocated to any pebble. Because we have allocated links to pebbles in levels t down to $t-b+1$, we consider the leftmost link allocatable to the pebble at level $t-b$. If we set m in the above calculation to $t-b$, then the leftmost link allocatable to the pebble at level $t-b$ is at most the (b^{t-b})-th one from the rightmost link in a sub-section. Substracting from the available free links (1), we get $b^{t-1} - b^{t-2} - b^{t-b}$, which is nonnegative if $b \geq 3$.

Using the above lemmas, it is easy to prove the following theorem.

Theorem 1. *Given a hash chain of length n, the amount of hash function evaluation m, and $k = m + 1$, the algorithm traverses the hash chain using m hash function evaluations at each iteration and using $k(\sqrt[k]{n} - 1)$ memory cells to store the intermediate hash values.*

4 Conclusion

Given the hash chain and a computation limit m ($k = m + 1$ and $b = \sqrt[k]{n}$), we have proposed an algorithm that traverses the hash chain using a total of

$k(b-1)$ memory. This reduces the memory requirements of Sella's algorithm by a factor of $(b-1)/b$. Because efficient hash chain traversal algorithms are aimed at devices with severely restricted computation and memory requirements, this reduction is considered to be important. Further, our algorithm is matches the memory requirements of Jakobsson's algorithm while still remaining scalable. Sella's algorithm, when scaled to the case of Jakobsson's algorithm, has a memory requirement of about twice that of Jakobsson's.

Acknowledgment. The author thanks the anonymous refrees for their kind and helpful comments.

References

1. D. Coppersmith and M. Jakobsson, Almost optimal hash sequence traversal, *Proc. of the Fifth Conference on Financial Cryptography (FC) '02*, Mar, 2002.
2. H. Haller, The S/Key one-time password system, *RFC 1760*, Internet Engineering Taskforce, Feb. 1995.
3. G. Itkis and L. Reyzin, Forward-secure signature with optimal signing and verifying, *Proc. of Crypto '01*, 332–354, 2001.
4. M. Jakobsson, Fractal hash sequence representation and traversal, *IEEE International Symposium on Information Theory (ISIT) 2002*, Lausanne, Switzerland, 2002.
5. A. Perrig, R. Canetti, D.Song, and D. Tygar, Efficient authentication and signing of multicast streams over lossy channels, *Proc. of IEEE Security and Provacy Symposium*, 56–73, May 2000.
6. A. Perrig, R. Canetti, D.Song, and D. Tygar, TESLA: Multicast source authentication transform, *Proposed IRFT Draft*, http://paris.cs.berkeley.edu/ perrig/
7. Yaron Sella, On the computation-storage trade-offs of hash chain traversal, *Proc. of the Sixth Conference on Financial Cryptography (FC) '03*, Mar, 2003.

Round Optimal Distributed Key Generation of Threshold Cryptosystem Based on Discrete Logarithm Problem

Rui Zhang and Hideki Imai

Information & Systems, Institute of Industrial Science, University of Tokyo
4-6-1, Komaba, Meguro-ku, Tokyo, 153-8505, Japan
zhang@imailab.iis.u-tokyo.ac.jp, imai@iis.u-tokyo.ac.jp

Abstract. We propose a distributed key generation protocol for discrete logarithm problem based threshold cryptosystems by introducing an efficient (publicly) verifiable encryption scheme from any homomorphic encryption with a non-interactive proof of fairness. Previous constructions of the same kind are either only based on a narrow definition of homomorphism or only a unique encryption scheme is considered. Our construction generalizes the scope of such design to a broader range of encryption schemes with efficient constructions of proofs of fairness. Since the protocol is round optimal (one-round) in the distributed fashion, adaptive adversary is not different from a static adversary, thus a simplified protocol design is possible. Our scheme is extremely capable for an environment with already built public key infrastructure. The verifiable encryption with fairness developed here can be used as building blocks of a variety of cryptographical applications like publicly verifiable secret sharing (PVSS), e-voting and auction schemes.

1 Introduction

A threshold cryptosystem distributes secret information among a set S of servers to build a fault-tolerant system, where there is a common public key pk and a number of secret key shares $\{sk_i\}_{i\in S}$ held by players in the group respectively. The players can cooperate to reconstruct the secret key, or even without it, sign messages or decrypt a ciphertext encrypted by pk, so that partial malfunctioning of total users will not affect the correctness of the output. A great proportion of solutions to multiparty protocols turns out to be a crux of threshold cryptosystem scheme in constructing a distributed TTP: key recovery [24], signature escrow [15,23], contract signing [27], fair exchange of digital signatures [2], e-voting [9,19] and auction [6] schemes.

One of the toughest tasks in constructing such fault-tolerant systems is the distributed key generation. A centralized server acting as a Trusted Party often becomes the target of attackers. A distributed key generation, i.e., each server contributes to the random secret key, is more important and desirable. Previous schemes of such threshold key generation have often adopted a multiparty-computation-fashioned protocol to distribute secret key shares among players

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 96–110, 2003.

against so-called adaptive adversaries. However, these schemes need intricate design techniques with inborn complexity in communication rounds: in a sharing phase the shares are sent to share holders via fully connected pair-wise secure channels and an additional complaining phase is needed to detect cheaters among a dealer and share holders via a broadcast channel. Shares' secrecy can be protected by secure channels, however, the cheater somehow survives even after the key generation, when some player accuses the dealer, a third player cannot distinguish between these two facts:

- The dealer has distributed a false share.
- The share holder lies on a good share.

In this paper, we try to solve this problem by proposing a non-interactive key generation under general computational assumption, where the share holder has no chance to misbehave.

1.1 Related Work

Threshold cryptosystems have been studied in [10,11,31,8,22], one common property of these schemes is that they work only with an honest majority. In [10, 11] Desmedt et al. define the concept of threshold schemes and propose the first implementation. However, this yields no provably secure solution. The first provable secure practical scheme is proposed by Shoup and Gennaro in [31] based on the Diffie-Hellman problem [12] in the Random Oracle model [13,4], which is resilient against a static adversary. Later research has focused on stronger attack model: in [21], Lysyanskaya et al. propose two new models of security for this kind of attacks namely the concurrent adversary and the rushing adversary. They further improve the result resisting an erasure-free adversary for signature schemes and encryption scheme [22], in which (1) an adversary corrupts servers at any point of the protocol depending on its entire view of the computation; (2) when corrupted, the players have to hand over the adversary their entire computation history; i.e., nothing can be erased.

The first distributed key generation of discrete logarithm problem based threshold cryptosystems is proposed in [26] by Pedersen, but later in [16] a flaw of their scheme has been found by Gennaro et al. that the key is not uniformly generated in the key space with some adversary and also a solution is given to deal with such type of adversary. In [8], Canetti et al. improve their results to resist adaptive adversary.

Fouque and Stern has noticed that a one-round key sharing protocol [14] can make the adaptive adversary meaningless, in the sense that it achieves nothing more than what a static adversary achieves. They have equipped a non-interactive proof of verifiable encryption discrete logarithm in the Paillier cryptosystem [25] in place of a complaining phase of key generation. However, in their scheme one can do nothing but choose Paillier cryptosystem as the only tool to construct the secret channel. On the other hand, to distribute discrete logarithm problem based key shares, one must manage keys of extra Paillier

cryptosystems, which may be based on composite degree residuosity assumptions: these degrade the security of the whole system by introducing with new keys of public key cryptosystems under different mathematical assumptions, resulting in a more challenging key management.

1.2 Our Results

We shall focus on one round distributed key generation of discrete logarithm based threshold schemes. Compared to previous schemes, our scheme enjoys following features:

Optimistic execution. First, we note that the servers should not always malfunction. In fact, in a real implementation, we can expect a much optimistic scenario. Previous solutions often make use of both broadcast channels and private channels in sharing phase, however, we notice that a private channel hides the adversary in such a way that if there should arouse a dispute a third party cannot tell whether the sender or the receiver is lying. So a special complaining phase is needed to correctly share a secret without a trusted dealer [26].

Theoretical results in [18,5,28] have already shown the possibility that given fully connected pair-wise secure channels, any multiparty computation (including the distributed key generation) can be performed provided that the number of dishonest players $t < n/3$ and given a broadcast channel together with pairwise private channels, can be secure against a static adversary corrupts up to $t < n/2$ players. However, complexity of their protocols prevents their application in practical use. Besides, since we are to construct a public key scheme, a private channel that guarantees information-theoretic secrecy is too luxury. A broadcast channel plus a public key encryption cryptosystem with "checkability" seems enough for this task. The concurrent adversary and adaptive erasure-free adversary as defined by [21] is of no significance in a one-round non-interactive protocol and a rushing attack will make no harm if we take the advantage of a synchronous network.

Robustness and efficiency. Our protocol provides security against dishonest majority cheaters, namely $t < n$, because the cheaters can be identified and eliminated in the end, while previous multiple-round protocols can work only with $t < n/2$ in the key generation phase thus in the whole protocol. Moreover, there is only a broadcast channel necessary while both broadcast channel and pairwise secret channel are used in [26,8]. We emphasize on the efficiency of our protocol. There are totally n^2 shares and each user broadcasts $O(ln)$ data where l is the security parameter, the whole data flow over the network is $O(ln^2)$. Considering the hidden coefficient in previous schemes and multiple rounds of communication, the complexity is acceptable for a network with low density and it can be further improved in an optimistic sense by using batch encryption techniques. A basic hypothesis here is that in practice a decryption server which stores the secret key share can process a large amount of parallel computations while the communication costs can be neglected.

User preference and flexibility on keys. Since each server may have already set up their encryption keys, introduction of additional Paillier cryptosystem may increase the amount of secret keys in the whole system. Our scheme works best for a situation that every player has his preference on keys. In previous schemes, if the public-secret key pairs of Paillier encryption have not been set up, there must be another auxiliary protocol executed to do the overhead. In our scheme, users have more flexibility.

Generalized design techniques. Another contribution of this work is a generalized template for designing non-interactive protocols. If a multiparty protocol can be simulated as information transmitted from a trusted party to all players then it can be designed using our template. The technique introduced here can be applied to other multiparty protocols as building blocks.

Smallest set of assumptions. A theoretical result of our protocol actually implies that a distributed discrete logarithm key generation protocol can only be based on discrete logarithm problem and existence of a broadcast channel, while previous protocols rely more assumptions beyond, like additional pairwise private channels [26] or other mathematical problem, e.g., deterministic residuosity [25, 14].

2 Preliminary

2.1 Basic Models

There are mainly two types of threshold cryptosystems in practice, namely, those based on the RSA problem and those on the discrete logarithm problem. We shall limit our scope to threshold cryptosystems based on intractability of the discrete logarithm problem because RSA based distributed key generation cannot be made one-round since a primality test is necessary for secret prime factors of public modulus, while discrete logarithm type is more amiable for the selection of secret keys. Let p and q denote large primes such that q divides $p-1$, Z_q is a subgroup of Z_p^* of order q, the secret key is an element x, such that public key is $y = g^x \bmod p$. Also in self-evident context, we also write $x = \log_g^y \bmod p$ as $x = \mathrm{DL}(y)$.

Network and players. We shall assume a synchronous network in all our discussion. Synchronous network can be constructed with GPS (Global Positioning System) [20] in practice. The players in the protocol should be connected with a broadcast channel which sends the same information to every player. There is no need to maintain any pairwise secure channels.

2.2 Security Definitions

Distributed secret key generation of threshold schemes. Assume n players over a synchronous open network are fully connected. A (t, n) distributed

threshold secret key generation is a protocol run among these players, who each chooses a secret number and shares it among all the players. Finally, honest players can decide a unique public key and corresponding secret key shares, such that if $t + 1$ or more players decrypt a ciphertext or issue signatures jointly, while t or less players can computationally have no information on the secret keys assuming that the discrete logarithm problem is intractable.

Possible adversarial behaviors. We follow [22] to describe the capabilities of the adversary. We assume the existence of n parties communicating over a synchronous broadcast channel with rushing, where up to a threshold l parties may be corrupted. Lysyanskaya et al in [22] describe a scheme to deal with such adversary, however, their protocol require interaction in the *Share* phase. An adaptive adversary can depend on the protocol history to schedule his attack while static adversary cannot. An *adaptive* adversary makes no difference from a static adversary if the protocol has only one round. Moreover, the underlying encryption scheme used here need not have chosen ciphertext security. Throughout this paper, the adversary is considered to be passive, whose attack strategy can be assumed to be fixed at the beginning of the protocol.

In our protocol, we permit any adversarial behavior trying to deviate the protocol from its correct execution, namely, an adversary may do any of the following:

- The adversary chooses to at most t players to corrupt. Once corrupted, the players have to hand all their data to the adversary and all their actions are controlled by the adversary.
- Each player chooses a random number and shares it using a public verifiable encryption via broadcast channel, while corrupted player may response correctly or not.
- The adversary can launch pre-scheduled attack during the execution, make arbitrary deviations from the protocol.

Security of distributed secret key generation for threshold cryptosystem. The security of the distributed secret key generation protocol is defined in terms of *correctness* and *confidentiality*. By correctness, we mean the following:

1. Each subset of at least $t + 1$ shares defines the same secret key x. t is called the threshold.
2. All honest parties have the same public key $y = g^x$. This should not be disturbed by a malicious adversary.
3. The secret key is uniformly distributed in the key space. That is, every element of subgroup will be chosen as the secret key with equivalent probability.

Confidentiality property requires that any PPT adversary should not learn any information on the secret key by corrupting at most t players. Furthermore, this can be expressed by simulatability. If a PPT adversary corrupts at most t servers which is erasure-free during the key generation, then the view of the adversary in the real execution shouldn't be distinguishable from the output of a program called a simulator, which is executed in expected polynomial time.

2.3 Homomorphic Encryption Schemes

Definition 1 (Public key encryption). A public key encryption scheme is a 3-tuple algorithm: (K, E, D), where K is the probabilistic key generation algorithm; E, maybe probabilistic, is the encryption algorithm which maps a plaintext m to a ciphertext c; D is a deterministic algorithm, which on input c recovers m.

Definition 2 (Homomorphic encryption). A function $F : G \to H$ maps from group G to group H is called a *homomorphism* of G into H if F preserves the operation of G. If $*$ and $\circ$ are polynomial time computable operations of G and H, respectively, then F preserves the operation of G if for all $a, b \in G$: $F(a * b) = F(a) \circ E(b)$. If an encryption scheme maps plaintext and ciphertext as above relations, we call it a homomorphic encryption scheme.

Remark 1. A homomorphic encryption must be malleable, which will not be IND-CCA secure [3]: given $E(M)$, the adversary can compute easily $E(M*k)$. We emphasize here that we are designing a non-interactive protocol, all encryption will be only be performed once, then discussions on adaptive security (chosen ciphertext security) can be omitted.

3 Previous Schemes

In this section, we briefly review some related protocols designed for distributed discrete logarithm key generation.

3.1 Pedersen's Scheme

Pedersen's scheme was the first attempt to eliminate the trusted dealer to construct a threshold key cryptosystem with shares generated from Shamir's secret sharing scheme [30]. He ingeniously used the homomorphic property of discrete logarithm. Each player acts as the dealer to use a broadcast channel to distribute his public key shares and public checksum and pair-wise private channels are used to distribute the secret share, finally the broadcast channel is used again to identify possible misbehavior.

A player P_i chooses $x_i \in_R Z_q$ at random. He also picks t random numbers $a_{i,k}$ in Z_q. For $f_i(X) = \sum_{k=0}^{t} a_{i,k} X^k$ where $a_{i,0} = x_i$, he privately sends a secret share $s_{i,j} = f_i(j) \bmod q$ to another player P_j with the private channel and broadcasts public information $y_{i,j} (= g^{s_{i,j}} \bmod p)$ and $A_{i,k} (= g^{a_{i,k}} \bmod p)$ for $k = 0, 1, .., t$. Each player P_j receives his secret share and the signature of P_i on the share and check if:

$$y_{i,j} = \prod_{k=0}^{t} A_{i,k}^{j^k} \bmod p$$

If $s_{i,j}$ is not the discrete logarithm of $y_{i,j}$ participant P_j broadcasts a complaint against P_i. If more than $n/2$ players complain against player P_i, then it is clear that player P_i is cheating and he will be disqualified. Otherwise, if less than $n/2$ players complained, P_i will reveal the secret key share $s_{i,j}$ and the signature on it for player P_j who has made the complaint. If he fails to reply any of the equation $y_{i,j} = g^{s_{i,j}}$, he will be disqualified. Otherwise he can still be qualified. Later a set of qualified players Q can be decided. The public key will be $y = \prod_{i \in Q} y_i$, where $y_i = A_{i,0} = y^{x_i}$.

Since after the complaining phase revealment of the secret key shares are performed, a flaw was indicated in [16] that the adversary can create a bias in the distribution of the last bit of the public key with probability 3/4 rather than 1/2. A second complaining phase is necessary for the identification of the cheaters and all players have to manage both a broadcast channel and private channels. However, this scheme is simple, no need to maintain other secret key of different cryptosystems.

3.2 Fouque and Stern's Scheme

By noticing the aftereffect of the private channel, Fouque and Stern propose a scheme using only public channel [14]: p'_j, q'_j are large primes and $N_j = p'_j q'_j$ is the modulus in Paillier cryptosystem. G_j is an element in $Z^*_{n^2}$of order a multiple of N_j. Let λ to be the Carmichael lambda fucntion $\lambda(N)$. The public key is $PK_j = (N, G)$ and the secret key is $SK_j = \lambda$.

For a non-interactive proof of fair encryption of discrete logarithm, Player P_i generates a random secret $s_{i,0}$, sets $a_{i,0} = s_{i,0}$ and chooses $a_{i,k}$ at random from Z_q for $1 \leq k \leq t$. Number $a_{i,0}, ..., a_{i,t}$ jointly define the polynomial $f_i(X) = \sum_{k=0}^{t} a_{i,k} X^k \in Z_p[X]$. Then he computes $s_{i,j} = f_i(j) \bmod p$. He broadcasts: for $k = 0, ..., t, A_{i,k} = g^{a_{i,k}} \bmod p$ and $y_{i,j} = g^{s_{i,j}} \bmod p$, $Y_{i,j} = G^{s_{i,j}} u_{i,j}^{N_j} \bmod N_j^2$, and a proof $(e_{i,j}, z_{i,j}, w_{i,j}) \in [0, B[\times[0, A[\times[0, N_j[$, where $e_{i,j} = H(g, G, y_{i,j}, Y_{i,j}, g^r y_{i,j}^{-e_{i,j}} \bmod p, G^{z_{i,j}} w^{N_j} Y_{i,j}^{-e_{i,j}} \bmod N_j^2)$, and $y_{i,j}^q = 1 \bmod p$, where $|A| \geq |B| \circ |S| + k$, k is the security parameter.

For each $1 \leq i, j \leq n$, the players verify that:

$$\prod_{k=0}^{t} A_{i,k}^{j^k} = \prod_{k=0}^{t} g^{\sum_{k=o}^{t} a_{i,k} j^k} = g^{f_i(j)} \bmod p$$

and check whether $g^{f_i(j)} \bmod p$ is equal to $y_{i,j}$ in order to verify that the distribution is correct. The players also verify the proofs $(e_{i,j}, w_{i,j}, z_{i,j})$: if $e = H(g, G, y_{i,j}, Y_{i,j}, g^{z_{i,j}} g^{-e_{i,j}} \bmod p, G^{z_{i,j}} w^{N_j} Y_{i,j}^{-e_{i,j}} \bmod N_j^2)$, and $y_{i,j}^p = 1 \bmod p$ for $1 \leq i, j \leq n$.

The players which do not follow the protocol are disqualified. The set of qualified players decide the public key y in the same way as in [26]. And the players will have correct secret key shares distributed. The public key will be the computed using Lagrangian interpolation polynomial. Thus secret key shares of discrete logarithm has been generated for a (t, n) threshold cryptosystem.

Remark 2. In fact, the proof system used in Fouque and Stern scheme based on Paillier encryption scheme is a specially "good" kind of homomorphic encryption schemes where $E(M_1 + M_2) = E(M_1) \times E(M_2)$ and $E(M_1 \times k) = E(M_1)^k$. An earlier work can even be found in Schnorr signature scheme [29]. We argue that e.g. the Nacacche-Stern encryption scheme and Okamoto-Uchiyama encryption scheme holding similar property and proof system can be formed analogously. However, not all the encryption schemes have such ideal properties, which may appear likely to be the following (e.g. RSA): $E(M_1 \times M_2) = E(M_1) \times E(M_2)$ *and* $E(M_1^k) = E(M_1)^k$.

4 Publicly Verifiable Encryption with Proof of Fairness

4.1 A One-Round Model for Threshold Cryptosystem

A trick is played here: a tag is attached to the ciphertext which allows everyone be able to check the validity of the ciphertext without decrypting it. Via a public channel, each player broadcasts a public commitment and verifiable encryption of all his shares. The secret key share to player P_j is encrypted under P_j's public key together with a tag so that everyone may check the correctness of the secret share while having no idea of what it is. So a cheating dealer can be detected and disqualified immediately in the key generation phase. If there is one honest player, the resulting public key and secret key will be distributed uniformly in the key space.

The construction of such a tag can in fact using general zero-knowledge for $\mathcal{NP}$ language combined with any encryption scheme to provide the encrypted plaintext is in fact the discrete logarithm of a publicly known value. However, to emphasize the efficiency of practical use, we construct a verifiable encryption of discrete logarithm over any homomorphic encryption schemes. Note that our definition of homomorphism is much broader.

4.2 Publicly Verifiable Encryption of Discrete Logarithm

The key idea of our publicly verifiable encryption is in fact a non-interactive publicly verifiable encryption, where with a public input y a prover proves a ciphertext c encrypts x under a public key PK (typically belongs to a third party), such that $(x, y) \in R$, where R is a binary relation varying from applications. In the discrete logarithm key generation, R refers to the discrete logarithm, i.e., x is the discrete logarithm of a public value y to the base g. Our protocol differs from the one in [7] is that our underlying encryption scheme is homomorphic, while the one in [7], used in a verifiable signature escrow context, should avoid homomorphic property to maintain chosen ciphertext security. Our scheme apparently may be more efficient than theirs. Abe in [1] has also designed checkable encryption schemes based on various underlying encryption in the random oracle model. However, the construction there is aiming at chosen ciphertext security and actually only proves whether the ciphertext sender knows the corresponding

plaintext. In our key generation protocol, we require not only that the ciphertext is formed validly, but also it should be discrete logarithm of some publicly known value, in other words, the relation R should be verified at the same time. Let's fist look at some examples with typical encryption schemes:

Example 1: ElGamal encryption. This scheme is observed to first appear in [32]. It is a combination of ElGamal encryption scheme and Schnorr signature assuming double exponent exists in a subgroup of prime order. The system assumes to be secure if DDH problem is hard in the subgroup. Let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that in (p, q, g) the DDH problem is hard. Here g is a generator of the group defined by p of prime order q.

- **Key generation**: $(p, q, h, z) \leftarrow \mathcal{G}(1^k)$, where h is a generator of Z_q in Z_q^*, $z \in_R Z_p$. Then the secret key is z and the public key is $v(= h^z \bmod p), p, q$ and a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$ (l is the security parameter, where 2^{-l} is negligible).
- **Encryption**: To verifiably encrypt x satisfying $y = g^x$, set $C = (A, B)$ as standard ElGamal encryption, where $a \in_R Z_q$, $A = h^a$, $B = x^{-1} \cdot v^a$, a proof of fairness can be constructed as follows: select $r_1, ..., r_l \in_R Z_q$, for $1 \leq i \leq l$, compute $w_{1i} = h^{r_i} \bmod p$ and $w_{2i} = g^{v^{r_i}}$. Denote $c = H(v, h, g, p, q, A, B, w_{11}, w_{21}, ..., w_{1l}, w_{2l})$.
- **Validity test**: Verifier checks if $c \stackrel{?}{=} H(v, h, g, p, q, A, B, w_{11}, w_{21}, ..., w_{1l}, w_{2l})$. Let c_i denotes the ith bit of $c = H(v, h, g, p, q, A, B, w_{11}, w_{21}, ..., w_{1l}, w_{2l})$, then for $1 \leq i \leq l$, $b_i = r_i - c_i \cdot a$, check if $w_{i1} \stackrel{?}{=} h^{b_i} A^{c_i}$, and $w_{i2} \stackrel{?}{=} g^{(v^{b_i})}$ if $c_i = 0$, or $w_{i2} \stackrel{?}{=} y^{(B \cdot v^{r_i})}$ if $c_i = 1$.
- **Decryption**: For a valid ciphertext that passes the above test, output $x = A^z/B$ as plaintext.

Example 2: RSA encryption. RSA encryption operates in the group Z_N^*, where $N = p'q'$, p' and q' are large primes. It is trivial that p', q', p, q must be distinct to maintain the system secure.

- **Key generation**: $(p', q') \leftarrow \mathcal{G}(1^k)$, $N = p'q'$, and $\phi(N) = (p'-1)(q'-1)$, so that Z_p is a subgroup of Z_N^* and Z_q is a subgroup of Z_p^* of order q. e is a small prime and select private key $d = e^{-1} \bmod \phi(N)$. Publish public key as e, N and a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$ (l is the security parameter, where 2^{-l} is negligible) as public encryption key.
- **Encryption**: $B = x^e \bmod N$, $y = g^x$, a proof of fairness can be constructed as follows: $r_1, ..., r_l \in_R Z_q$, for $0 \leq i \leq l$, $A_i = r_i^e \bmod N$. let c_i denote ith bit of $c = H(e, N, B, A_1, A_2, ..., A_l, b_1, ..., b_l)$, $b_i = x^{c_i} \cdot r_l \bmod N$. Then the ciphertext is B and the proof is $(A_1, ..., A_l, b_1, ..., b_l)$.
- **Validity test**: First check if $c \stackrel{?}{=} H(e, N, B, A_1, A_2, ..., A_l, b_1, ..., b_l)$. For $1 \leq i \leq l$, let c_i denote the ith bit of $c = H(e, N, B, A_1, A_2, ..., A_l, b_1, ..., b_l)$, verify $b_i^e \stackrel{?}{=} A_i$ if $c_i = 0$, or $b_i = B \cdot A_i$ if $c_i = 1$.

– **Decryption**: For a valid ciphertext that passes the above test, output $x = c^d \bmod N$ as plaintext.

Remark 3. In these schemes the total data size broadcast expands with a factor of l, the security parameter, however, the probability error is exponentially small (2^{-l}) in these construction. Note these are not to perform frequently.

5 Generalization to Arbitrary Homomorphic Encryption

In this section, we show the above results can be generalized to arbitrary homomorphic encryption / decryption schemes. As shown above, any encryption scheme that maps a plaintext from the input domain G to a ciphertext in the output domain H preserving operations can be expressed as:

$$E(M_1 * M_2) = E(M_1) \circ E(M_2)$$

Without loss of generality, assume $*$, $\circ$, $\star$ are three operations (e.g. $+$, $\times$, exponentiation in finite field) defined in ascending order of priority in both G and H. For a homomorphic encryption maps preserved operations in G and in H, the homomorphic properties can be written as:

$$* \to \circ \tag{1}$$
$$\circ \to \star \tag{2}$$
$$\circ \to \circ \tag{3}$$
$$\star \to \star \tag{4}$$

For example, (1) shows that $*$ operation in G (plaintext space) is preserved as $*$ in H (ciphertext space). Since encryption scheme must at least be oneway for it must be infeasible for anyone to infer the ciphertext to the plaintext, then above mappings must hold oneway, which infers that it is possible to open some masked value on the right side while maintaining secret on the left side. For the relation to prove and underlying encryption scheme: either the same operation preserved in G and H, or an operation in G by relation $\mathcal{R}$ is preserved in H by the encryption scheme. If two such pairs can be found to hold for the relation and the encryption scheme, a proof system can then be constructed.

Case 1: The same operation is preserved in the domain for the relation $\mathcal{R}$ and the encryption E, e.g., the pair (1,2) for which $\circ$ is preserved in both G and H holds for both the relation and the homomorphic encryption scheme. The proof system can be constructed efficiently using Fiat-Shamir heuristic [13]. In such case, a challenge of the first homomorphism relation can be easily turned to the second one, then a zero-knowledge proof can be constructed proving the $R(x, y)$ under PK.

Case 2: The same operation is preserved in different domains G and H respectively. E.g. (1,2) for the relation and (3,4) for the homomorphic encryption,

which form the proof system for verifiable RSA encryption of discrete logarithm in Example 2. In such case since there is no direct way bridging over the challenge of the first one to the second one, however, a cut-and-choose methodology can be executed sequentially by mapping (1) (with respect to (2)) in G to (3) (with respect to (4)) in H. In such cases, the error probability is exponentially small after sequential repeated executions.

Generally, we have the following theorem on Case 1:

Theorem 1. *Proofs from Case 1 based on homomorphic encryption can be made zero-knowledge efficiently.*

Proof. We shall first make the construction and then prove that it satisfies the requirements of zero-knowledge. For definitions and precise model of zero-knowledge proof please refer to [17]. There are three operations in all. For relation $\mathcal{R}$, assume $M_1 * M_2 \to M_1 \circ M_2$, $M_1 \circ k \to M_1 \star k$ and homomorphic encryption schemes: $E(M_1 * M_2) = E(M_1) \circ E(M_2)$ and $E(M_1 \circ k) = E(M_1) \star k$. The prover chooses a random element r from G and compute $E(r)$ in H as a commitment. Such commitment by the encryption scheme is unconditionally binding (guaranteed by deterministic property of a public key encryption scheme) and computationally concealing. Then from a public random source (for simplicity, consider a random oracle), a random reference c is derived and the prover uses the two pairs of relations, $E(r * x) \to E(r) \circ E(x)$ and $E(c \circ x) \to E(x) \star c$. He then sends $w = r * x \circ c$, $E(r)$, $E(x)$ to the verifier, who verifies the two pairs of relations: by $E(w) = E(r) \circ E(x) \star c$.

First, it is easily seen that an honest prover will always be able to construct such proof. Second, assume that a cheating prover doesn't know x and is able to compute such proofs. Since c is a public random source accessible by both the prover and verifier, in other words, c is fixed once chosen, then having the ability to find $x' \neq x$ from the output $E(r * x' \circ c) = E(r * x \circ c)$ contradicts the deterministic property of the underlying assumption of the encryption schemes. Third, given w and $E(x)$, the verifier can easily simulates all the information gained in the protocol, by obtaining a random c' from the random public accessible source, calculate x' from $w' = r' * x' \circ c'$ and construct $E(r' * x' \circ c') = E(r') \circ E(x') \star c'$.

Case 2 is a little cumbersome, since no direct operation is preserved in both encryption scheme and fairness of an relation (here discrete logarithm of a public value). Now the relation (here for example discrete logarithm) maps $M_1 * M_2 \to M_1 \circ M_2$, however, the homomorphic encryption only maps $E(M_1 \circ M_2)$ to $E(M_1) \circ E(M_2)$ and $E(M_1 \star k)$ to $E(M_1) \star k$. Since there is no direct method to obtain $E(M_1) \star k$ in general, we cannot proceed except for some special cases where an efficient operation can be achieved for one bit (e.g. homomorphic relation $M \star 0 \to 1$ or $M \star 1 \to M$ hold for discrete logarithm), we can then base the proof on cut-and-choose methodology.

For $1 \leq i \leq l$, the prover chooses r_i randomly from G. Let the public random source output a challenge bit c_i. The prover computes $E(r_i)$ and $w_i = r_i * x \circ c_i$, and together with $E(x)$ sends them to the verifier. The verifier opens every commitment by verifying $E(w_i) = E(r_i) \circ E(x) \star 0 = E(r_i)$, if $c_i = 0$;

$E(w_i) = E(r_i) \circ E(x) \star 1 = E(r_i) \circ E(x)$ if $c_i = 1$. Fortunately, most of finite field arithmetics satisfy this principle. We base our construction on the following theorem:

Theorem 2. *The construction for Case 2 is a secure zero-knowledge proof with error probability exponentially small.*

Proof. Again, the commitment by using public key encryption is again unconditionally binding and computationally concealing. First, completeness is clear that an honest prover can construct such proof. Second, this is a standard "cut-and-choose" argument, that an honest verifier can only be cheated is roughly 2^{-l}, if the l-bit string is chosen uniformly from the public random source. Third, for $1 \leq i \leq l$, let the verifier constructs a simulator outputs random r'_i and forms the proof in the same way $E(r'_i * x \circ c'_i)$, where c'_i is the ith random bit obtained from the public random source. In the view of an adversary whether this is from an honest prover or the simulator is computationally indistinguishable, if $E(r_i)$ and $E(r'_i)$ are identically distributed in H.

Since RSA encryption has the homomorphic encryption property of Case 2, we can further prove its security with similar reasoning:

Corollary 1. *The protocol given in Example 2 is a secure verifiable encryption with proof of fairness, where public randomness is generated by random oracle.*

6 Distributed Key Generation for Discrete Logarithm

6.1 The Scheme

The main protocol uses homomorphic encryption with public verifiable proof of fairness as a building block to build a (t, n) threshold cryptosystem. To have public key shares pk_i of player P_i and corresponding secret key shares $(sk_1, ..., sk_n)$ correctly distributed, suppose each player P_j has his personal public key and secret key pair (E_{PK_j}, D_{SK_j}) (they can be chosen independently and based on different assumptions). Each share is verifiably encrypted and broadcasted to every player so that every player can verify if other players have received the correct share. A cheating dealer can be caught immediately once he cheats. Since the network is supposed to be synchronous, then rushing attacks and other attacks will not take place. The main protocol is performed as follows:

1. For $1 \leq i \leq n$, P_i chooses a random number x_i from the $x_i \in_R Z_p$, computes $y_i = g^{x_i}$, and share x_i with Shamir's secret sharing scheme: he sets: $a_{i,0} = s_{i,0} = x_i$ and chooses $a_{i,k}$ at random from Z_q for $1 \leq k \leq t$, which the number $a_{i,0}, ..., a_{i,t}$ define a polynomial $f_i(X) = \sum_{k=0}^{t} a_{i,k} X^k \in Z_q[X]$ of degree t. Then he computes $s_{i,j} = f_i(j) \bmod p$. He broadcasts: for $k = 0, ..., t$, $A_{i,k} = g^{a_{i,k}} \bmod p$ and $y_{i,j} = g^{s_{i,j}} \bmod p$, and an encryption $E_{PK_j}(s_{i,j})$ of secret key share for Player P_j under the correct publicly verifiable encryption with proof of fairness introduced in section 5. Especially he will keep the share when $j = i$.

2. Player $P_1, ..., P_n$ each verifies

$$\prod_{k=0}^{t} A_{i,k}^{j^k} = \prod_{k=0}^{t} (g^{a_{i,k}})^{j^k} = g^{\sum_{k=o}^{t} a_{i,k} j^k} = g^{f_i(j)}$$

and check whether $g^{f_i(j)}$ is equal to $y_{i,j}$ in order to verify that the distribution is correct. The players also verify the proofs that $E_{PK_j}(s_{i,j})$ is the correct encryption of $s_{i,j}$ to the public key PK_j.

3. The players which do not follow the protocol will be disqualified. The remained set of player forms the set Q and they can generate the threshold key system with the public key $(y, y_1, ..., y_n)$ and secret shares $(x_1, ..., x_n)$ where Player P_j will get correct shares from $i \in$ Q, and compute the following:

$$x_j = \sum_{j \in Q} s_{i,j}, \;\; y_j = g^{x_j}, \;\; y = \prod_{i \in Q} A_{i,0} = g^{\sum_{i \in Q} f_i(0)} \qquad i, j \in \mathrm{Q}$$

respectively, $A_{i,0}$ can be computed from the Lagrangian interpolation polynomial given $A_{i,j}$ $(1 \leq j \leq t+1)$.

4. The secret can be constructed if $t+1$ honest players gather. $sk = \sum_{j=1}^{t+1} b_i x_i$, where $b_i = \prod_{1 \leq k \leq t+1, k \neq j} \frac{j}{k-j}$.

6.2 Security Analysis

We get the following theorem on analysis of our protocol:

Theorem 3. *Our scheme is a secure distributed key generation protocol for a threshold cryptosystem.*

Proof. We build our proof on the following two claims for two requirements in the security definitions:

Claim. Our protocol is correct against passive adversary.

It is clear that any subset of $t+1$ honest players can reconstruct the secret key. For the public verifiability, the honest players can be decided, thus the public key can be uniquely decided. If there should be one honest player, the public key y and the secret key x will be uniformly random in the subgroup.

Claim. Our protocol protects the confidentiality of the secret key.

For the zero-knowledge property of the publicly verifiable encryption scheme, a PPT adversary cannot break the confidentiality of the secret shares. Furthermore, We can build a simulator which is a pre-decided program in stead of real protocol run. In the view of an adversary, the sharing phase can simulated: it is computationally indistinguishable for the adversary to tell the output from a simulator from a real program. Since the verifiable encryption is simulatable.

Combining two claims, we complete the proof of the theorem.

Remark 4. Furthermore, the scheme in [14] can be regarded as a special case if we take the proof knowledge of the style in the Pailler encryption for Case 1.

7 Conclusion

Our key generation protocol is round optimal (one round). Our non-interactive encryption with proof of fairness can be used as building blocks for other complicated schemes. We argue that actually, the technique developed here can be used to any scenario where data flow from the Trusted Authority to the players without interactions. Since homomorphic encryption is malleable, it is not possibly to achieve security against an adaptive adversary under multiple use. However, it is possible to be made *composable* if an independent set up with randomly chosen parameters is applied every time.

References

1. M. Abe. Securing "encryption + proof of knowledge" in the random oracle model. In *CT-RSA 2002*, volume 2271 of *LNCS*, pages 277–289. Springer-Verlag, 2002.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, volume 1462 of *LNCS*. Spring-Verlag, 1998.
4. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
5. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC'88. ACM, 1988.
6. C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *6th ACM Conference on Computer and Communications Security (CCS)*, pages 120–127. ACM, 1999.
7. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT'00*, volume 1976 of *LNCS*, pages 331–345. Springer-Verlag, 2000.
8. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115. Springer-Verlag, 99.
9. R. Cramer, M. Franklin, B. Schoenmakers, , and M. Yung. Multi-authority secret ballot elections with linear work. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 72–83. Springer-Verlag, 1999.
10. Y. Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO'87*, volume 293 of *LNCS*, pages 120–127. Springer-Verlag, 1987.
11. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer-Verlag, 1990.
12. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
13. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problem. In *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1987.

14. P. Fouque and J. Stern. One round threshold discrete-log key generation without private channels. In *PKC'01*, volume 1992 of *LNCS*, pages 300–316. Springer-Verlag, 2001.
15. M. Franklin and M. Reiter. Verifiable signature sharing. In *EUROCRYPT'95*, volume 921 of *LNCS*, pages 50–63. Springer-Verlag, 1995.
16. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310. Springer-Verlag, 1999.
17. O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
18. O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *19th STOC*, pages 25–27. Springer-Verlag, 1987.
19. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 539–556. Springer-Verlag, 2000.
20. `http://tycho.usno.navy.mil/gps.html`.
21. S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 221–242. Springer-Verlag, 2000.
22. A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *ASIACRYPT'01*, volume 2248 of *LNCS*, pages 331–350. Springer-Verlag, 2001.
23. W. Mao. Verifiable escrowed signature. In *ACISP*, volume 1270 of *LNCS*, pages 240–248. Springer-Verlag, 1997.
24. S. Micali. Fair public-key cryptosystems. In *CRYPTO'92*, volume 740 of *LNCS*, pages 113–138. Springer-Verlag, 1993.
25. P. Paillier. Public key cryptosystem based on composite degree residuosity classes. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.
26. T.P. Pedersen. A threshold cryptosystem without a trusted party. In *Eurocrypt'91*, volume 547 of *LNCS*, pages 522–526. Springer-Verlag, 1991.
27. B. Pfitzmann, M. Schunter, and M. Waidner. Optimal efficiency of optimistic contract signing. In *17th PODC*, pages 113–122. Springer-Verlag, 1998.
28. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *21th STOC*, LNCS, pages 73–85. Springer-Verlag, 1989.
29. C.P. Schnorr. Ecient signature generation for smart cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer-Verlag, 1990.
30. A. Shamir. How to share a secret. *Communications of ACM*, 22:612–613, 1979.
31. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attacks. In *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 1–16. Springer-Verlag, 1998.
32. M. Stadler. Publicly verifiable secret sharing. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 190–199. Springer-Verlag, 1996.

On the Security of Two Threshold Signature Schemes with Traceable Signers

Guilin Wang[1], Xiaoxi Han[1,2], and Bo Zhu[1]

[1] Infocomm Security Department
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
http://www.i2r.a-star.edu.sg/icsd/
{glwang,xiaoxi,zhubo}@i2r.a-star.edu.sg
[2] Laboratory of Computer Science, Institute of Software
Chinese Academy of Sciences, Beijing 100080, P.R. China

Abstract. A (t, n) threshold signature scheme allows t or more group members to generate signatures on behalf of a group with n members, while any $t-1$ or less members cannot do the same thing. In 2001, based on a variant of ElGamal digital signature scheme, Li et al. proposed two (t, n) threshold signature schemes with traceable signers. One of their schemes needs the assistance of a mutually trusted center, while the other does not. In this paper, we present a security analysis on their schemes. We first point out that in fact signers in their schemes are *untraceable*, since anybody can convert a valid threshold signature into a new one such that another subset of group members will be wrongly considered as the signers of the new threshold signature for the same message. Furthermore, we demonstrate an attack to show that their second threshold signature scheme is *insecure*. In our attack, $(n - t + 1)$ colluding members can control the group secret key. Therefore, they can generate valid threshold signature for any message without the help of other members. Furthermore, honest members cannot detect this security flaw in the system, since any t members can generate threshold signatures according to the prescribed protocols.

Keywords: digital signature, threshold signature, cryptanalysis.

1 Introduction

In 1987, Desmedt first introduced the concept of *group-orinted cryptography* [3]. In contrast to traditional cryptosystems, in a group-oriented cryptosystem, only several collaborative entities can perform a cryptographic operation (e.g. encryption, decryption, generation or verification of signatures.). *Threshold signature* is one such system, in which only t or more group members can generate signatures on behalf of a group with n members, while any $t-1$ or less members cannot do the same thing. On the other hand, to check the validity of a threshold signature, a verifier only needs to know the unique group public key. According to whether a verifier can trace back the signers of a threshold signature, there

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 111–122, 2003.

are two kinds of threshold signature schemes: with anonymous signers and with traceable signers. According to whether a trusted center is involved, threshold signature schemes can be classified into two types: with or without a trusted center. A scheme without the need of a trusted center is also called a distributed threshold signature scheme.

Based on RSA system, Desmedt and Frankel proposed the first threshold signature scheme in [4]. Gennaro et al. presented efficient threshold DSS signature schemes and threshold RSA signature schemes in [6] and [7] respectively. Stinson and Strobl proposed a provably secure distributed threshold Schnorr signature scheme and used it to design implicit certificates [12]. All these schemes do not provide the property to trace the identities of signers of a threshold signature.

In 2001, Li et al. proposed a variant of ElGamal digital signature scheme [5,9], and then based on this ElGamal type signature scheme, they constructed two (t, n) threshold signature schemes with traceable signers [10]: One of their schemes needs the assistance of a mutually trusted center, while the other does not. To show that their schemes are secure, they examined that several known attacks cannot be mounted in their schemes. However, we find that in fact their schemes cannot be used to trace the signers of a signature, and that their second scheme (i.e., the distributed one) is insecure.

More specifically, in this paper we first point out that the threshold signatures in their schemes are *untraceable* since anybody can convert a valid threshold signature into a new one such that another set of group members becomes the signers of the new threshold signature for the same message. Moreover, we demonstrate an attack to show that in their second threshold signature scheme, $(n - t + 1)$ colluding members can control the group secret key. Therefore, they can generate valid threshold signature for any message without the help of other members. Furthermore, honest members cannot detect any security flaw in the system, since under the assumption that the clerk is also corrupted, any t members can generate threshold signatures according to the prescribed protocols. In addition, we provide some countermeasures to prevent our attack.

The rest of this paper is organized as follows. Section 2 introduces Li et al.'s modified ElGamal signature scheme on which their threshold signature schemes are based. Section 3 reviews Li et al.'s second threshold signature scheme, and Section 4 presents our security analysis on their schemes. The conclusion is drawn in Section 5.

2 Modified ElGamal Signature Scheme

In this section, we review Li et al.'s modified ElGamal signature scheme [10] that is developed from generalized ElGamal digital signature schemes [9]. Li et al.'s threshold signature schemes are based on this modified ElGamal signature scheme.

System parameters:

- p: a large prime number such that $(p - 1)/2$ is also a prime number;
- g: a primitive element of the finite field $GF(p)$;

- $H(\cdot)$: a one-way hash function;
- x_u: the secret key of a user U;
- y_u: the related public key of the user U where $y_u = g^{x_u} \bmod p$.

Signature Generation. To sign a message m, a user U randomly chooses a number $k \in [1, p-1]$ and computes:

$$r = g^k \bmod p. \tag{1}$$

and then solves the following equation for s:

$$rs = (r + H(m))k + x_u \bmod p - 1. \tag{2}$$

The signature on the message m is (r, s).

Signature Verification. To verify a signature (r, s) on a message m, one checks whether the following equality holds:

$$g^{rs} \equiv r^{r+H(m)} y_u \bmod p. \tag{3}$$

3 Review of Li et al.'s Threshold Signature Schemes

In [10], Li et al. proposed two (t, n) threshold signature schemes with traceable signers: The first one needs a mutually trusted center while the second one does not. However, the methods of tracing back in these two schemes are the same. In this section we only review their second scheme which does not need a trusted center.

Their second scheme consists of seven parts: system parameter setup, group public key generation, individual signature generation, individual signature verification, threshold signature generation, threshold signature verification, and signer identification.

Part 1. System parameter setup

The following public system parameters are agreed by all group members.

- p, q: two large prime numbers such that $q|(p-1)$;
- g: a generator of order q in finite field $GF(p)$;
- $H(\cdot)$: a one-way hash function.

Part 2. Group public key generation

Let U_i, $i \in A = \{1, 2, \cdots, n\}$, be n group members in the system. For simplicity, we say A is the set of all group members, and similarly we denote a subset of group members by B, where $B \subset A$. To generate the group public key, each member U_i ($i \in A$) randomly selects his secret key x_i and a public identity ID_i. Then, he publishes his public key $y_i = g^{x_i} \bmod p$. When all y_i's ($i \in A$) are available, the group public key y is set by

$$y = \prod_{i \in A} y_i \bmod p.$$

Now, each group member uses Shamir's $(t, n-1)$ secret sharing scheme [11] to distribute his secret key to other $n-1$ members. That is, member U_i first selects a random $(t-1)$-degree polynomial $f_i(X)$ with $f_i(0) = x_i \bmod q$, sends the secret shadow $f_i(ID_j) \bmod q$ to each member U_j privately, and then publishes the related public key $y_{ij} = g^{f_i(ID_j)} \bmod p$ as public information, where $j \in A$ and $j \neq i$.

Part 3. Individual signature generation

When any t members of group A want to generate a signature for a message m, each member U_i, $i \in B$ ($B \subseteq A$ and $|B| = t$), selects a random integer $k_i \in [1, q-1]$, computes and broadcasts $r_i = g^{k_i} \bmod p$. After all r_i's are available, the following value R is calculated:

$$R = \prod_{i \in B} r_i \bmod p.$$

Using his secret key x_i, secret shadows $f_j(ID_i) \bmod q$ ($j \in A \setminus B$) and k_i, the group member U_i computes s_i from the following equation:

$$s_i R = (R + H(m))k_i + x_i + \sum_{j \in A \setminus B} f_j(ID_i) \cdot C_{Bi} \bmod q. \tag{4}$$

In the above equation, C_{Bi} is the Lagrange interpolating coefficient given by

$$C_{Bi} = \prod_{k \in B \setminus \{i\}} \frac{ID_k}{ID_k - ID_i} \bmod q. \tag{5}$$

Finally, each member U_i ($i \in B$) sends his individual signature (r_i, s_i) to a designated clerk.

Part 4. Individual signature verification

On receiving the individual signature (r_i, s_i) for message m from member U_i, the clerk uses public information y_i and y_{ij} ($j \in A \setminus B$) to verify the validity of (r_i, s_i) by

$$g^{s_i R} \equiv r_i^{R+H(m)} y_i \Big(\prod_{j \in A \setminus B} y_{ji} \Big)^{C_{Bi}} \bmod p. \tag{6}$$

If the above equation holds, the individual signature (r_i, s_i) from U_i is valid. At the same time, the clerk uses t pairs of public values (ID_i, y_i) ($i \in B$) to construct a Lagrange polynomial function $h(Y)$ as follows:

$$h(Y) = \sum_{i \in B} \Big(ID_i \prod_{j \in B \setminus \{i\}} \frac{Y - y_j}{y_i - y_j} \Big) \bmod q. \tag{7}$$

Later, the function $h(Y)$ will be used to trace the signers who signed the threshold signature for m.

Part 5. Threshold signature generation

After t individual signatures are received and validated, the threshold signature (R, S) for the message m is computed by

$$R = \prod_{i \in B} r_i \bmod p, \quad \text{and} \quad S = \sum_{i \in B} s_i \bmod q. \tag{8}$$

Part 6. Threshold signature verification

Any outsider can use the group public key y to check the validity of a threshold signature (R, S) for a message m by

$$g^{SR} \equiv R^{R+H(m)} y \bmod p. \tag{9}$$

If the above equation holds, the threshold signature (R, S) is valid. Otherwise, it is invalid.

Part 7. Signer identification

The authors of [10] implicitly assumed that the function $h(Y)$ is attached to the signature pair (R, S). Therefore, when a verifier wants to know the signers of a threshold signature (R, S) for a message m, he uses public values (ID_i, y_i) $(i \in A)$ to find all signers by the following equation:

$$ID_i \stackrel{?}{\equiv} h(y_i). \tag{10}$$

If the above equation holds, the group member U_i is one signer of the signature (R, S). Otherwise, he is not.

4 On the Security of Li et al.'s Schemes

In this section, we discuss the security of Li et al.'s threshold signature schemes. We first analyze the traceability of their two schemes and point out that in fact both of them are untraceable. Then, we demonstrate an attack on their second threshold signature scheme which does not need a trusted center. Finally, some improvements are also provided to prevent our attack.

4.1 On the Traceability of Li et al.'s Schemes

The methods of tracing back in their two threshold signature schemes are the same. That is, using public values (ID_i, y_i) and the function $h(Y)$ to determine whether Equation (10) holds. According to Equation (7), however, we know that the function $h(Y)$ is determined by public values, and that there is no intrinsic relationship between a signature pair (R, S) and $h(Y)$ since the same pair (R, S) for the message m may be generated by any t group members. Therefore, given a valid pair (R, S) for a message m, anybody can construct a function $h(Y)$ such that any t members, $U_{i_1}, \cdots, U_{i_t}$, could be the signers of (R, S). In addition, even if the values of ID_i and y_i are known only by group members, Li et al.'s

schemes are also untraceable since we have the following simple attack. Given two threshold signature pairs $(R, S, h(Y))$ and $(R', S', h'(Y))$ for two messages m and m', it is easy to see that $(R, S, h'(Y))$ and $(R', S', h(Y))$ are also two valid threshold signature pairs for messages m and m', respectively.

From the above discussion, it is clear that the signer tracing method proposed in [10] does not work. Thus, in Li et al.'s two schemes the signers of a threshold signature are untraceable, instead of traceable.

4.2 An Attack on Li et al.'s Second Scheme

In this subsection, we demonstrate an attack on Li et al.'s second threshold signature scheme in which n group members generate the group public key in a distributed way. We present the details about how $(n-t+1)$ colluding members can cheat other $(t-1)$ honest members by controlling the group secret key. In our attack, we make the following two assumptions:

- **Assumption 1.** Except $(t-1)$ honest members in the system, all other $(n-t+1)$ members are dishonest and collude together.
- **Assumption 2.** The designated clerk is also corrupted by dishonest members.

After our attack is presented, we will discuss why these two assumptions are necessary and why they are reasonable in applications. For simplicity, but without loss of generality, we assume that the first $(t-1)$ members, i.e., $U_1, \cdots, U_{t-1}$, are honest members and all other $(n-t+1)$ members are dishonest. We further assume that as the head of dishonest members, U_n controls the group secret key, while other colluding members tolerate his faults in the group public key generation.

It is obvious that, using the group secret key, anyone of these $(n-t+1)$ malicious members can forge a valid threshold signature on any message independently. Furthermore, we demonstrate that if $n \in B$ or $B = \{1, 2, \cdots, t-1, j\}$ where $j \in \{t, \cdots, n-1\}$, then under the help of the corrupted clerk, the t members coming from B can generate valid individual signatures and threshold signatures. Moreover, even in the case where $|\bar{B} \cap \{t, t+1, \cdots, n-1\}| \geq 2$, the t members in $\bar{B}$ also can generate valid threshold signatures but one malicious member in $\bar{B}$ cannot generate valid individual signatures. For example, if $t = 8$ and $n = 10$, we know that in a secure threshold signature scheme, a valid threshold signature can only be generated by 8 or more group members. However, in Li et al.'s second scheme our attack enables members U_8, U_9 and U_{10} to control the group secret key x and to cheat other seven honest members. By using the known group secret key x, any of U_8, U_9 or U_{10} can independently generate valid threshold signature for any message. At the same time, under the help of the corrupted clerk, any eight out of ten members can generate valid threshold signatures. Furthermore, except in the case where the signers consist of U_8, U_9 and six out of seven honest members, colluding members also can provide valid individual signatures.

We emphasize that the above property is important in real world, since it allows dishonest members to cheat honest members that the system is normal

and secure until a forged threshold signature on a message appears. Otherwise, even if U_n controls the group secret key but any t members cannot generate a threshold signature by the prescribed protocols, honest members will soon doubt the security and usability of the system.

The details of our attack are given as follows. The whole procedure consists of three steps: member U_n controlling the group private key, member U_n distributing secret shares, and dishonest members generating valid individual signatures.

Step 1. Member U_n controlling the group private key

In the group public key generation of Li et al.'s second scheme, it is not required that all public keys y_i's should be published simultaneously. Thus, member U_n can be the last one to publish his public key y_n. By choosing a random number x as the group secret key, he first sets the group public key by $y = g^x \bmod p$. Then, when all other y_i's ($i \in \{1, \cdots, n-1\}$) are published, he computes and publishes his public key y_n as follows:

$$y_n = y \cdot \prod_{i=1}^{n-1} y_i^{-1} \bmod p.$$

Hence, all members in group A will take y as the group public key, since the following equation holds:

$$y = y_n \cdot \prod_{i=1}^{n-1} y_i = g^x \bmod p.$$

Therefore, member U_n has controlled the group private key x corresponding to y. Of course, member U_n does not know his private key x_n corresponding to y_n since he cannot find the discrete logarithm of y_n to the base g.

Step 2. Member U_n distributing secret shares

The difficulty is how member U_n can distribute his secret key x_n to other members even though he does not know the value of x_n. For this sake, U_n does as follows.

1. Firstly, U_n assumes that he has chosen a $(t-1)$-degree polynomial $f_n(X)$ such that $f_n(0) = x_n$, where x_n is the unknown but fixed number satisfying $y_n = g^{x_n} \bmod p$. At the same time, he selects $(t-1)$ random numbers $b_i \in [0, q-1]$, and sets $f_n(ID_i) = b_i$, for each $i \in \{1, 2, \cdots, t-1\}$.
2. Secondly, he sends $f_n(ID_i)$ privately to each member U_i, $i \in \{1, 2, \cdots, t-1\}$. However, U_n cannot send $f_n(ID_j)$ to each of his conspirators since he does not know the value of $f_n(ID_j)$ for each $j \in \{t, t+1, \cdots, n-1\}$. But, as U_n's conspirators, each member U_j tolerates this fault.
3. Thirdly, U_n has to publish the related public key y_{nj}, for all $j \in \{1, 2, \cdots, n-1\}$. Of course, for $i \in \{1, 2, \cdots, t-1\}$, U_n computes y_{ni} by $y_{ni} = g^{b_i} \bmod p$. Moreover, we now explain that U_n can also carry out $y_{nj} = g^{f_n(ID_j)}$ for each $j \in \{t, t+1, \cdots, n-1\}$ even though he does not know the value of

$f_n(ID_j)$. According to the Lagrange interpolating formula, the following equation holds:

$$y_n = g^{f_n(0)} = g^{C_{B_j j} \cdot f_n(ID_j)} \cdot \prod_{i=1}^{t-1} g^{C_{B_j i} \cdot f_n(ID_i)} \bmod p, \quad \forall\, j \in \{t, t+1, \cdots, n-1\}.$$

In the above equation, $B_j = \{1, 2, \cdots, t-1, j\}$, $C_{B_j j}$ is the Lagrange interpolating coefficient given by Equation (5), and $f_n(ID_i) = b_i$. Therefore, U_n can get the value of $y_{nj} = g^{f_n(ID_j)} \bmod p$ by the following equation:

$$y_{nj} = (y_n \cdot \prod_{i=1}^{t-1} g^{-C_{B_j i} \cdot b_i})^{C_{B_j j}^{-1}} \bmod p, \quad \forall j \in \{t, t+1, \cdots, n-1\}. \tag{11}$$

After all y_{nj} ($j \in \{1, 2, \cdots, n-1\}$) are computed, U_n publishes them. Any member can verify that all y_{nj}'s are consistent since the following equation holds:

$$y_n \equiv \prod_{j \in B} y_{nj}^{C_{Bj}} \bmod p, \quad \forall B \subseteq \{1, 2, \cdots, n-1\} \text{ and } |B| = t. \tag{12}$$

When U_n and all other members published all y_{ij}, $i, j \in \{1, 2, \cdots, n\}$ and $i \neq j$, the system is set up. After that, if necessary, any dishonest member can use the known group secret key x to forge valid threshold signature on any message m. That is, he first chooses a random $k \in [0, q-1]$ and computes $R = g^k \bmod p$. Then, he gets S from equation $RS = (R + H(m))k + x \bmod p$. It is easy to know that such forged pair (R, S) is a valid threshold signature on message m, since it satisfies Equation (9). Furthermore, as we have mentioned, under the help of the corrupted clerk, dishonest members can also generate valid individual signature. So they can cheat honest members that the system is normal and secure.

Step 3. Dishonest members generating their individual signatures

Now, we assume t members of a subset B want to sign a message m. According to the individual signature generation Equation (4), U_n cannot generate valid individual signature since he does not know the value of x_n. In addition, if $n \notin B$, any malicious member U_j ($t \leq j \leq n-1$) cannot generate valid individual signature since he does not know the value of $f_n(ID_j)$. However, note that if $n \in B$, any U_j ($t \leq j \leq n-1$) can generate his individual signature. Therefore, under our assumption 2, we will see that the corrupted clerk can help dishonest members to generate valid individual signatures in two cases: (1) $n \in B$ and (2) $B = \{1, 2, \cdots, t-1, j\}$ where $j \in \{t, t+1, \cdots, n-1\}$. In the following, we only describe how dishonest members can generate their valid individual signatures in case 1. As we mentioned above, in this case $n \in B$, any other (honest or dishonest) member can generate his individual signature normally. Therefore, we now focus on how member U_n can generate his individual signature.

In Li et al.'s scheme, it is also not required that all r_i's should be published simultaneously, thereby member U_n can be the last one to publish r_n. That is, U_n first selects a random number $k \in [0, q-1]$, and sets

$$R = g^k \bmod p. \tag{13}$$

When all other r_i's have been broadcast, he computes and broadcasts the following r_n:

$$r_n = R \cdot \prod_{i \in B/\{n\}} r_i^{-1} \bmod p.$$

Consequently, each member U_j ($j \in B/\{n\}$) computes R by $R = r_n \cdot \prod_{i \in B/\{n\}} r_i \bmod p$ ($= g^k \bmod p$). Then, by using Equation (4), each U_j generates and sends his individual signature (r_i, s_i) to the clerk. The corrupted clerk reveals the values of all (r_i, s_i)'s to U_n. To get his individual signature on the message m, U_n first solves S from the following equation

$$SR = (R + H(m))k + x \bmod q. \tag{14}$$

Next, he computes his individual signature (r_n, s_n) as follows.

$$r_n = R \prod_{i \in B \setminus \{n\}} r_i^{-1} \bmod q, \quad \text{and} \quad s_n = S - \sum_{i \in B \setminus \{n\}} s_i \bmod q. \tag{15}$$

Finally, U_n sends (r_n, s_n) to the clerk so that the clerk can generate the threshold signature (R, S) for the message m. If necessary, the clerk publishes all individual signatures (r_i, s_i) ($i \in B$) as the evidences that all members in B indeed generated valid individual signatures for the message m. After all (r_i, s_i)'s have been broadcast, each member in B can verify the validity of each pair (r_i, s_i) by using Equation (6). Up to this piont, U_n generated his individual signature pair (r_n, s_n).

The following theorem proves that the above (R, S) is a valid threshold signature for the message m, and that (r_n, s_n) is U_n's valid individual signature for the message m.

Theorem 1. *The above procedure that U_n generates his individual signature is successful. That is,*

(1) The pair (R, S) generated by Equation (14) is a valid threshold signature for the message m, and
(2) The pair (r_n, s_n) generated by Equation (15) is U_n's valid individual signature for the same message m.

Proof: (1) It is obvious that the pair (R, S) generated by Equation (14) satisfies Equation (9). We now prove (2): i.e., we need to show that the pair (r_n, s_n) generated by Equation (15) satisfies Equation (6). This is justified by the following equalities.

$$\begin{aligned}
g^{s_n R} &= g^{(S - \sum_{i \in B/\{n\}} s_i)R} \bmod p \\
&= g^{(R+H(m))k} \cdot y \cdot \left[\prod_{i \in B/\{n\}} \left(r_i^{R+H(m)} \cdot y_i \cdot (\prod_{j \in A/B} y_{ji})^{C_{Bi}} \right) \right]^{-1} \bmod p \\
&= \left(R \prod_{i \in B/\{n\}} r_i^{-1} \right)^{R+H(m)} \cdot y \prod_{i \in B/\{n\}} y_i^{-1} \cdot (\prod_{j \in A/B} \prod_{i \in B/\{n\}} y_{ji}^{-C_{Bi}}) \\
&= r_n^{R+H(m)} \cdot y \prod_{i \in B/\{n\}} y_i^{-1} \cdot \prod_{j \in A/B} y_{jn}^{C_{Bn}} \cdot \prod_{j \in A/B} (\prod_{i \in B} y_{ji}^{-C_{Bi}}) \bmod p \\
&= r_n^{R+H(m)} \cdot y \prod_{i \in B/\{n\}} y_i^{-1} \cdot \prod_{j \in A/B} y_{jn}^{C_{Bn}} \cdot \prod_{j \in A/B} y_j^{-1} \bmod p \\
&= r_n^{R+H(m)} \cdot y_n \cdot (\prod_{j \in A/B} y_{jn})^{C_{Bn}} \bmod p.
\end{aligned}$$

When any U_j ($j \in \{t, t+1, \cdots, n-1\}$) and $t-1$ honest members want to generate a threshold signature, U_j can generate his valid individual signature analogously, and similar result as Theorem 1 can also be proved.

4.3 Remarks and Improvements

In this subsection, we give some remarks on our attack and the two assumptions listed in the beginning of Section 4.2. In addition, several improvements on Li et al.'s second scheme are provided to prevent the above attack.

We first comment that our attack is stronger in the sense that it allows malicious members to provide valid individual signatures. Actually, in the Li et al.'s original second scheme, only the clerk checks the validity of individual signatures. In this case, t members in any subset B can generate valid threshold signature as follows. A malicious member U_j in B first controls the value of R as in the foregoing attack. He then solves S from equation $SR = (R + H(m))k + x \bmod q$, and sends the threshold signature pair (R, S) to the corrupted clerk.

Now, we point out that the success of our attack does not depend on whether the number of dishonest members $(n-t+1)$ is less than t. However, of course, our attack has practical meanings only in the case where $(n-t+1) < t$, since the basic assumption of all threshold cryptosystems is that there are at most $(t-1)$ malicious members. Therefore, our assumption 1, $(n-t+1)$ members colluding together, is reasonable in application settings.

We remark that our assumption 2 is also reasonable in many applications. In Li et al.'s second scheme, the clerk only performs some computations that anyone can do, and the computational result, i.e. the threshold signature (R, S), is publicly verifiable. In a practical system, it is unlikely to set the clerk as a trusted party. Therefore, dishonest members can corrupt and make use of him for a long time, since the clerk is a designated entity. Furthermore, a natural invariant of Li et al.' second scheme is to remove the clerk by requiring each member to broadcast his individual signature (r_i, s_i). In this case, we do not need assumption 2 any more. At the same time, dishonest members can generate their individual signatures by taking the same steps as in section 4.2.

From the foregoing description, we know that our attack results from the insecurity of the group public key generation protocol in Li et al.'s second scheme. Therefore, to avoid our attack, each member should publish his public key y_i simultaneously. One simple way is to require that all members first commit their y_i's and then open their commitments to reveal y_i's. Another choice is to require that each member proves his knowledge of the discrete logarithm of y_i to the base g by using interactive or non-interactive knowledge proof protocols [1,2]. The third choice is to use Gennaro et al.'s distributed key generation protocol for discrete-log cryptosystems [8] as the group public key generation protocol. At the same time, we know that in our attack dishonest members can use other members' r_i's and s_i's to generate their counterparts. Therefore, all members should first commit them before revealing their values.

5 Conclusion

In this paper, we presented a security analysis to Li et al.'s two (n, t) threshold signature schemes with traceable signers [10]. We first pointed out that in their schemes the signers of a signature are in fact untraceable, since anybody can convert a valid threshold signature into a new one for the same message. As a result of it, another subset of group members will be wrongly identified as the signers of the new threshold signature for the same message. Furthermore, on their scheme without a mutually trusted center, we demonstrated an attack that allows $(n - t + 1)$ colluding members to control the group secret key and then generate valid threshold signature for any message. However, honest members cannot detect this security flaw in the system since any t members can generate threshold signatures according to the specified protocols. Consequently, colluding dishonest members can cheat honest members successfully. In addition, some countermeasures are provided to prevent our attack.

References

1. J. Camenisch and M. Stadler. Effient group signature schemes for large groups. In: *Advances in Cryptology – CRYPTO'97, LNCS 1294*, pp. 410–424. Springer-Verlag, 1997.
2. D. Chaum and T.P. Pedersen. Wallet databases with observers. In: *Advances in Cryptology – CRYPTO'92, LNCS 740*, pp. 89–105. Springer-Verlag, 1993.
3. Y. Desmedt. Society and group oriented cryptography: A new concept. In: *Advances in Cryptology – CRYPTO'87, LNCS 293*, pp. 120–127. Springer-Verlag, 1988.
4. Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures (Extended Abstract). In: *Advances in Cryptology – CRYPTO'91, LNCS 576*, pp.457–469. Springer-Verlag, 1991.
5. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 1985, 31: 469–472.
6. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In: *Advances in Cryptology-EUROCRYPT'96, LNCS 1070*, pp. 354–371. Springer-Verlag, 1996. Also appears in *Information and Computation*, 2001, 164(1): 54–84.
7. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. In: *Advances in Cryptology – CRYPTO'96, LNCS 1109*, pp. 157–172. Springer-Verlag, 1996. Also appears in *Journal of Cryptology*, 2000, 13: 273–300.
8. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology - EUROCRYPT'99, LNCS 1592*, pp. 295–310. Springer-Verlag, 1999.
9. L. Harn and Y. Xu. Design of generalized ElGamal type digital signature schemes based on discrete logarithm. *Electronic Letters*, 1994, 24(31): 2025–2026.
10. Z.-C. Li, J.-M. Zhang, J. Luo, W. Song, and Y.-Q. Dai. Group-oriented (t, n) threshold digital signature schemes with traceable signers. In: *Topics in Electronic Commerce* (*ISEC 2001*), *LNCS 2040*, pp. 57–69. Springer-Verlag, 2001.

11. A. Shamir. How to share a secret. *Communications of the ACM*, 1979, 22(11): 612–613.
12. D.R. Stinson and R. Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In: *Information Security and Privacy* (*ACISP'01*), *LNCS 2119*, pp. 417–434. Springer-Verlag, 2001.

Proxy and Threshold One-Time Signatures

Mohamed Al-Ibrahim[1] and Anton Cerny[2]

[1] Center for Advanced Computing
Computing Department
Macquarie University
Sydney , NSW 2109, Australia
ibrahim@ieee.org
[2] Department of Mathematics and Computer Science
Kuwait University
P.O. Box 5969, Safat 13060, Kuwait
cerny@mcs.kuniv.edu.kw

Abstract. One-time signatures are an important and efficient authentication utility. Various schemes already exist for the classical one-way public-key cryptography. One-time signatures have not been sufficiently explored in the literature in the branch of society-oriented cryptography. Their particular properties make them suitable, as a potential cryptographic primitive, for broadcast communication and group-based applications. In this paper, we try to contribute to filling this gap by introducing several group-based one-time signature schemes of various versions: with proxy, with trusted party, and without trusted party.

Keywords: One-time signature, proxy signature, group signatures, digital signatures, threshold cryptography, broadcast authentication

1 Introduction

One-time signatures are an important public-key cryptography primitive. They derive their importance from their fast signature verification. This is in contrast to the conventional digital signature schemes, which usually have high generation or verification computation time. One-time signatures are an ideal option for authenticating particular types of applications where receivers are of low power capability, such as smart cards, or for online applications, such as video/audio streaming, which require fast verification.

One-time signatures are efficient and secure. Typically, signature parameters are initialized well ahead of the time when messages are to be signed and verified. This allows the signer to pre-compute the signature parameters so they can be fetched by potential verifiers. Once the message is known, the signer can sign it quickly, and receivers can verify the signed message efficiently. A distinct characteristic of one-time signatures is that they are used once only. To sign a new message, the signer must initialize the signature parameters. This means that the parameters of old signatures are not reused to sign another message;

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 123–136, 2003.

otherwise, the signature parameters would be exposed. The security of one-time signatures is measured by the difficulty of forging a signature by an adversary who normally has access to a single pair: a message and its signature.

On the other hand, groups play an important role in contemporary communication. Numerous examples of group applications include the stock exchange, collaborative tasks, and many other multicast applications. Group communication has a (potentially) high communication overhead, and it is desirable that group members can authenticate their communications efficiently. Cryptographic transformations by a group of participants was the subject of investigation in so called *society-oriented cryptography* ([6], [4]). Unlike classical cryptography, society-oriented cryptography allows groups of cooperating participants to carry out cryptographic transformations. That is, society-oriented cryptography requires distribution of the power of performing a cryptographic transformation among a group of participants such that only designated subsets of the group can perform the required cryptographic operation, but unauthorized subsets cannot do so.

With the recent interest in securing group and broadcast communication, there has been a great demand for designing a new class of fast signature schemes that can handle a vast number of signatures from broadcast or group-based applications efficiently, rather than using typical signature schemes. Hence, there have been a number of attempts in society-oriented cryptography to design signature schemes for authenticating group-based scenarios.

Several schemes were proposed that use classical authentication schemes such as digital signatures (RSA [24], ElGamal [7]) for group-based transformations. However, these conventional methods typically have a high computational overhead, and hence they may not fulfill the new requirements with regard to the efficiency of the emerging applications. Besides, the nature of authenticating online real-time applications usually requires a fast authentication. That is, the extra embedded complexity which is involved in the typical digital signatures to provide extra security adds more computational time. In contrast, one-time signatures provide the required security services with less computational overhead.

As mentioned, one-time signatures are potentially far more (computationally) efficient than classical authentication methods. This leads us to explore new ways of improving the efficiency of authentication in group communication using one-time signatures. That is, we attempt to apply one-time signatures for situations where the right to execute signature operations is shared by a group of signers. We propose a scheme for a threshold proxy signature. We achieve this by introducing a few intermediate schemes. We put together the concepts of one-time signature and threshold group signature by using the idea of proxy signing. Proxy one-time signature was first used in [12] to authenticate mobile agents in low-bandwidth communications. On the other hand, and to the best of our knowledge, the problem of finding a group oriented one-time signature has not been discussed elsewhere.

In this paper, we begin by reviewing the relevant work in the area of one-time signatures. To allow application of several one-time signature schemes in a

common way, we establish a construction model for the proposed protocols. To reach our goal, we investigate the problem of one-time signature in two scenarios: with a proxy signer and with a group of signers. We start with one-time signature for the case when a signer wants to delegate his one-time signature to a proxy who would sign on his behalf. Then in Section 5, we show how it is possible to construct a threshold one-time signature using the Shamir secret sharing method. This may happen with or without the aid of a trusted party. Finally, in Section 6, we design a scheme for threshold proxy signature.

2 Related Work

Lamport [13], Rabin [16], Merkle [17] and GMR [11] are well known examples of one-time signature schemes. They share the same basic idea and are based on committing to secret keys via one-way functions. Rabin uses an interactive approach for verification of signatures with the signer. These schemes differ in their approaches, but they share the same idea: only one message can be signed using the same key. Once the signature is released, its private key is not used again; otherwise, it is possible for an adversary to compute the secret key.

A new approach to designing such signatures is the BiBa one-time signature [19]. The BiBa signature exploits the birthday paradox property. A large number of secret keys is used to find collisions among the generated keys associated with the message. This way of signing requires a long pre-computational time. Reyzin and Reyzin [23] solve BiBa's disadvantage of having a very long signing time. Their idea is to calculate the number of required keys according to the size of the message and pre-determined length of the signature. Based on this, key generation would be very fast, and hence signing is faster.

One-time signatures have been used in group communication for authenticating streaming applications in multicast communication. Gennaro and Rohatchi [9] used a chained method with one-time signature. Rohatchi used a k-times one-time signature based on on-line/off-line approach. Perrig used it in Tesla [18]. Al-Ibrahim *et al.* in [1] introduced k-sibling one-time signature for authenticating transit flows.

3 A Class of One-Time Signature Schemes

In this section, we establish a model for one-time signature schemes. The model is not aimed at introducing a new kind of signature. We want to set a common view of several well-established signature schemes in order to be able to apply any one of them in our subsequent scenarios. Not every signature scheme in our model is therefore secure and the properties of each such particular scheme are to be investigated separately.

Our model consists of a signer S and a verifier V and is described as a tuple $\mathcal{O} = (M, X, Y, h, v, \pi)$ where M is the set of messages, X, Y are finite sets, $h : X \to Y$ is a one-way hash function, $v \geq 1$ is an integer and $\pi : M \to 2^{\{1,2,\cdots,v\}}$

is a function. All parts of $\mathcal{O}$ are public. If a signer S sends a message $m \in M$ to a verifier V, the signature creation and verification proceeds as follows:

Key generation

Signer S

1. chooses v random values $s_1, s_2, \ldots, s_v \in X$ (the secret keys of the signer)
2. computes v values $p_1 = h(s_1)$, $p_2 = h(s_2), \ldots, p_v = h(s_v) \in Y$ and makes them public.

Signing

Signer S

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. sends m and the signature $(s_{j_1}, s_{j_2}, \ldots, s_{j_r})$ to the verifier V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $h_1 = h(s_{j_1}), h_2 = h(s_{j_2}), \ldots, h_r = h(s_{j_r})$
3. accepts the signature if and only if $h_1 = p_{j_1}, h_2 = p_{j_2}, \ldots, h_r = p_{j_r}$.

The model includes schemes like Lamport [13] or Merkle [17] as special cases. The schemes of Rabin [22], GMR [11], BiBa [19] are of a different type. The "better than BiBa" scheme of Reyzin and Reyzin [23], Bos and Chaum [3], and HORS++ of [21] belong to this model.

4 A Simple One-Time Proxy Signature Scheme

Delegation of rights is a common practice in the real world. A manager of an institution may delegate to one of his deputies the capability to sign on behalf of the institution while he is on holiday. For electronic transactions, a similar approach is needed to delegate the manager digital signature to the deputy.

Proxy signature is a signature scheme where an original signer delegates his/her signing capability to a proxy signer, and then the proxy signer creates a signature on behalf of the original signer. When a receiver verifies a proxy signature, he verifies both the signature itself and the original signer's delegation. Mambo, Usuda and Okamoto (MUO) in [15] established models for proxy signatures. They classified proxy signatures, based on delegation type, as full delegation, partial delegation, and delegation by warrant. In full delegation, the signer gives his secret key to the proxy. In partial delegation, the signer creates a separate secret key for the proxy, but it is derived from his secret key. In signing with warrant, the signer signs the public key. In addition, they provide various constructions for proxy signature schemes with detailed security description and analysis. Their proxy signatures provide various security services including:

- **Unforgeability**. Only the proxy signer (besides the original signer) can create a valid signature for the original signer.
- **Proxy signer's deviation**. Each valid proxy signer's signature can be detected as her signature.
- **Verifiability**. A positive verification of a proxy's signature guarantees the agreement of the original signer with this signature.
- **Distinguishability**. A valid proxy's signature can be distinguished from the original signer's signature (in polynomial time).
- **Identifiability**. The original signer can determine the identity of a proxy from his signature (if there are more proxy signers).
- **Undeniabilty.** A proxy signer cannot disavow his valid signature.

Detailed discussion may be found in [15].

Zhang in [27] noticed that the Mambo scheme does not provide

- **Nonrepudiation**. Neither the original nor the proxy signer can falsely deny later that he generated a signature.

In [27] the scheme from [15] has been enhanced to provide nonrepudiation.

The scheme in [15] allows the proxy to sign an arbitrary number of messages. Furthermore, using the warrant is not a systematic way to control the number of signed messages in electronic communication. In some situations, the signer may need to delegate his signature to the proxy for one-time/one-purpose only. For example, an autocratic manager may want, for security or administrative reasons, to restrict the proxy to signing on his behalf for one time only. Hence, a new type of proxy signature that is more restricted than the Mambo approach is needed. An efficient one-time signature can be used in this case. Kim *et al.* in [12] designed a one-time proxy signature using fail-stop signature to provide authentication to mobile agents applications. In our proposal, we use a class of one-time signatures, as those described in section 3, to design a new one-time proxy signature.

If we consider a "classical" type of scheme, where the original signer shares his secret keys with the proxy or generates new secret keys for the proxy, distinguishability and non-repudiation are not guaranteed, since both the original and the proxy signer know the secret keys. The character of a one-time signature allows us to adopt a principally new approach, where the secret keys are generated and kept by the proxy only, and the original signer has no share in the secret. The original signer only confirms the public keys and stores them with a trusted party (registry). The security properties such as unforgeability, proxy signer's deviation, verifiability, and undeniability of the scheme are the same as in the underlying one-time signature scheme $\mathcal{O}$. Introducing the proxy signer clearly does not cause any violation of these security properties, unless the signature scheme is used more than once by an unreliable proxy. Signing several messages using the same set of keys reveals too many secret keys, and an eavesdropper could easily sign a false message using them. Our suggested solution

involves the trusted party. Let us assume that, besides the public keys approved by the original signer, one more value will be stored by the proxy when signing a message. However, this action will be guarded by the trusted party and will not be allowed to be done more than once. When signing the message m, the proxy will store there the value $h(m)$. In an additional step, the verifier will compute $h(m)$ for the received message m and check it against the value from the registry. Since this value can be stored to the registry just once, only one message can be legally signed. The scheme involves a signer S, a proxy P, a verifier V, and a trusted party TP. It uses the one-time signature scheme $\mathcal{O} = (M, X, Y, h, v, \pi)$ where X is a sufficiently large set. In addition, we assume that h is extended to $h : X \cup M \to Y$ while still preserving its one-wayness.

Key generation

Signer S

1. asks P to start generating secret keys.

Proxy P

1. chooses v secret numbers : $s_1, s_2, \ldots, s_v \in X$
2. computes $p_1 = h(s_1), p_2 = h(s_2), \ldots, p_v = h(s_v)$
3. passes $(p_1, p_2, \ldots, p_v)$ to S.

Signer S

1. verifies that the p's are from P (a one-time signature of P can be used for signing the list of p's)
2. makes $(p_1, p_2, \ldots, p_v)$ public, registered to the name of S.

Signing

Proxy P

1. computes $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $q = h(m)$ and registers this value with TP as a one-time writable value
3. sends $(m, s_{j_1}, \ldots, s_{j_r})$ to V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $h_1 = h(s_{j_1}), h_2 = h(s_{j_2}), \ldots, h_r = h(s_{j_r})$
3. computes $q' = h(m)$
4. fetches $p_{j_1}, p_{j_2}, \ldots, p_{j_r}$ and q from TP
5. accepts the signature if and only if $h_1 = p_{j_1}, h_2 = p_{j_2}, \ldots, h_r = p_{j_r}$ and $q' = q$.

The proxy uses its private keys, and the public keys are stored with a trusted party; hence, the proxy cannot deny signing or revealing the secret to a third agent - a danger occurring in most of the established proxy signature schemes. Since the signer and the proxy do not share the same key, non-repudiation is achieved. Sending keys by the proxy to the signer in the key generation phase does not compromise security since these keys will become public anyway. The role of the original signer is to endorse the public-keys generated by the proxy signer to the registry. This step is crucial; otherwise, any agent may claim itself to be a proxy for the original signer.

5 A (t, n) Threshold One-Time Signature Scheme

A particularly interesting class of society-oriented cryptography which includes threshold cryptographic transformation, consists of all subsets of t or more participants from a group of n members. A digital signature is an integer issued by a signer which depends on both the signer's secret key and the message to be signed. In conventional cryptosystems, the signer is a single user. However, the process of signing may need to be shared by a group of participants. The first attempts at designing a shared signature were made by Boyd. Threshold RSA [5] and Threshold ElGamal [14] signatures are examples of threshold multisignature schemes that require the presence of t participants of the group to sign a message. Both schemes exploit the Threshold Shamir secret sharing method to generate shares of signatures.

Here, we attempt to expand the idea of threshold signatures from the conventional cryptosystems transformations into one-time signatures to benefit from its efficiency properties in speeding-up the verification process. Our model consists of a group of signers $S_i, i = 1, 2, \ldots, n$ and a verifier V. A one-time signature scheme $\mathcal{O} = (M, \mathbb{F}, Y, h, v, \pi)$ is used, where $\mathbb{F}$ is a finite field and Y is a finite set, and both are sufficiently large. A threshold value $t \leq n$ is specified in advance. Not less than t signers are required to sign a message.

5.1 With a Trusted Party

In our first scenario, two more parties are involved: a trusted distributor D, and a trusted combiner C.

The idea behind this scheme is to let the distributor D choose the secret keys of the general one-time signature scheme of the group.

The shares of these secret keys for the particular signers are computed by D using the Shamir secret sharing algorithm, and they are then distributed to the participants. For each secret key s_j, a distinct polynomial f_j of degree $t-1$ with $f_j(0) = s_j$ is used to create secret shares. A public value x_i is associated with each signer S_i; his secret share on the key s_j is then $s_{i,j} = f_j(x_i)$. The set of polynomials comprising the system is illustrated (graphically) in Figure 1. Each intersection of a polynomial with the y axis represents a secret key. Two or more shares of the same signer may be identical, since several polynomials may have

a common value in some of the points x_i (the graphs may intersect on a vertical line at x_i). This clearly does not compromise the security of the system, since this information is known only to D and the the signer. The secret keys s_j are chosen to be pairwise distinct; hence no two polynomial graphs intersect on the y axis.

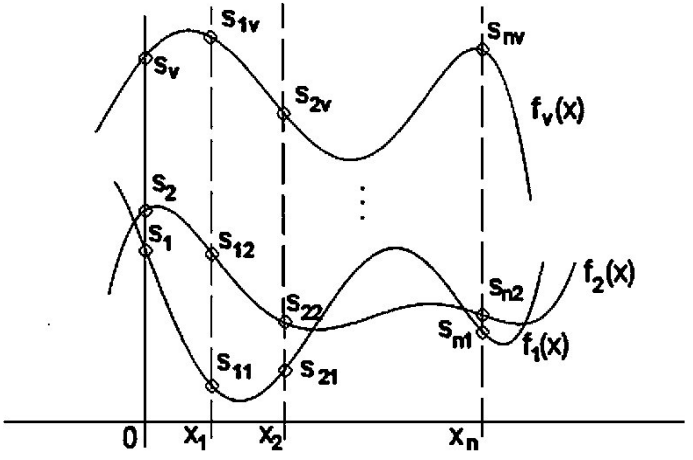

Fig. 1.

Key generation and share distribution *Distributor D*

1. chooses randomly v pairwise distinct elements $s_j \in \mathbb{F}, j = 1, \ldots, v$ (group secret keys)
2. computes the v values $p_j = h(s_j), j = 1, \ldots, v$ and makes them public (registered to the group name)
3. chooses randomly n pairwise distinct non-zero elements $x_i, i = 1, \ldots, n$ from $\mathbb{F}$ and makes them public
4. chooses randomly v polynomials $f_j(x) = f_{j,0} + f_{j,1}x + \cdots + f_{j,t-1}x^{t-1}, j = 1, \ldots, v$, satisfying $f_j(0) = f_{j,0} = s_j$
5. computes $s_{i,j} = f_j(x_i), i = 1, \ldots, n, j = 1, \ldots, v$
6. sends $(s_{i,j})_{j=1,\ldots,v}$ by a secure channel to the signer $S_i, i = 1, \ldots, n$ (secret share for the partial signer S_i)

Signing

Signer $S_i, i \in \{i_1, i_2, \ldots, i_t\}$

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. sends the partial signature $(s_{i,j_1}, s_{i,j_2}, \ldots, s_{i,j_r})$ to C.

Combiner C

1. waits to receive partial signatures from (at least) t signers $S_{i_1}, \ldots, S_{i_t}$
2. using Lagrange interpolation, recovers the polynomials $f_{j_k}(x)$, $k = 1, \ldots, r$, based on the conditions $f_{j_k}(x_{i_1}) = s_{i_1,j_k}, \ldots, f_{j_k}(x_{i_t}) = s_{i_t,j_k}$
3. finds $(s_{j_1}, s_{j_2}, \ldots, s_{j_r}) = (f_{j_1,0}, f_{j_2,0}, \ldots, f_{j_r,0})$
4. sends $(m, s_{j_1}, s_{j_2}, \ldots, s_{j_r})$ to V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. fetches p_{j_k}, $k = 1, \ldots, r$ from the registry
3. checks whether $p_{j_k} = h(s_{j_k})$, $k = 1, \ldots, r$.

Since the usual Shamir secret sharing is used, at least t signers are necessary to find any of the group secret keys. The fact that at most one message will be signed using the same signature may be guaranteed by the trusted combiner, so the multiple signature problem vanishes. In our scheme, the trusted distributor D knows all the secret keys; therefore, his reliability must be without doubt. The next version, without a trusted party, avoids such strict requirements. Observe that the combiner C knows only those secret keys which are used for the signature and which will be revealed to the verifier.

The computation of the shares involves nv times evaluation of a polynomial of degree $t-1$ by D and r times Lagrange interpolation of a polynomial of degree $t-1$ by C. In addition, D, V and each partial signer must compute $\pi(m)$ and D and V compute v and r values of the function h, respectively. The signers may compute π in parallel. It is worth noting that the verification of the one-time signature scheme is as efficient as without secret sharing.

5.2 Without a Trusted Party

The scenario without a trusted party works the same way as the one with the trusted party; the steps of the distributor D and combiner C are performed by the signers themselves. In particular, the set of n signers is used as a parallel n-processor computer.

Key generation and share distribution

Signer $S_i, i = 1, \ldots, n$

1. chooses randomly a non-zero element $x_i \in \mathbb{F}$ and makes (i, x_i) public
2. chooses randomly $s_j \in \mathbb{F}$ (secret key) for each $j = 1, \ldots, v$ such that $(j - 1) \bmod n + 1 = i$
3. computes the value $p_j = h(s_j)$ and makes the pair (j, p_j) public (registered to the group name)
4. chooses randomly a polynomial $f_j(x) = f_{j,0} + f_{j,1}x + \cdots + f_{j,t-1}x^{t-1}$ satisfying $f_j(0) = f_{j,0} = s_j$
5. computes $s_{i',j} = f_j(x_{i'}), i' = 1, \ldots, n$
6. sends $s_{i',j}$ by a secure channel to the signer $S_{i'}, i' = 1, \ldots, n, i' \neq i$ (secret share for the signer $S_{i'}$)

Signing

Signer $S_i, i \in \{i_1, i_2, \ldots, i_t\}$

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. sends the partial signature $(s_{i,j_1}, s_{i,j_2}, \ldots, s_{i,j_r})$: the triple (i, j_k, s_{i,j_k}) is sent to S_{i_q} where $q = (j_k - 1) \bmod t + 1, k = 1, 2, \ldots, r$
3. using Lagrange interpolation, based on the conditions $f_{j_k}(x_{i_1}) = s_{i_1,j_k}$, $f_{j_k}(x_{i_2}) = s_{i_2,j_k}, \ldots, f_{j_k}(x_{i_t}) = s_{i_t,j_k}$,recovers the polynomial $f_{j_k}(x)$ for each complete t-tuple $((i_1, j_k, s_{i_1,j_k}), \ldots, (i_t, j_k, s_{i_1,j_k}))$ received
4. for each polynomial f_j recovered, finds $s_j = f_{j,0}$
5. for each polynomial f_j recovered, sends (m, j, s_j) to V.

Verification

Verifier V

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. fetches p_{j_k}, $k = 1, \ldots, r$ from the registry
3. waits until all triples (m, j_k, s_{j_k}), $k = 1, \ldots, r$ are received
4. checks whether $p_{j_k} = h(s_{j_k})$, $k = 1, \ldots, r$.

Let r_0 be the minimum length of a signature and v the total number of secret keys. In our scheme, each of the n signers generates $\lceil v/n \rceil$ secret keys. We claim that if $\lceil v/n \rceil (t - 1) < r_0$, then at least t out of the n signers are necessary to create a valid signature. Indeed, $(t - 1)$ signers know at most $\lceil v/n \rceil (t - 1)$ secret keys. If the condition holds, this is not enough to create a signature of length r_0. On the other hand, the multiple signatures problem arises here again. Several messages may be signed even without malicious intention, since two independent subgroups of size t may sign two different messages. This problem may again be resolved by the "trusted registry" as in Section 4. Another solution may be an explicit synchronization of all signers before signing by one subgroup. That is, on time slot σ, all the n participants would be involved in signing the message m though there are only t actual signers. We leave open the problem of

designing a better scheme for one-time signatures that would prevent subgroups in a threshold scheme from signing more than one message at a time.

The complexity considerations from Part 5.1 are valid, except that, instead of the time necessary for computating nv polynomial values, the time for computing $\max(n, n\lceil v/n\rceil \approx v)$ values is required, since the signers may compute in parallel. In a similar way, only the time for $\lceil r/t\rceil$ Lagrange interpolations is necessary.

How realistic is the condition $\lceil v/n\rceil\,(t-1) < r_0$? If the scheme of Lamport ([13]) is used to sign a message of length μ, then $v = 2\mu$ are generated and the signature consists of μ keys. Our condition is satisfied if $t < n/2 + 1$.

6 A (t, n) Threshold Proxy One-Time Signature Scheme

In this section, we combine the ideas from the two previous sections and propose the following model. A group of n signers $S_i, i = 1, 2, \ldots, n$ wants to allow any subgroup of at least t signers to sign a message using a one-time signature. In our solution, the group will play the role of the original signer, who delegates his right to use a one-time signature to any subgroup of $t \le n$ (proxy). The signature is to be verified by a verifier V. A one-time signature scheme $\mathcal{O} = (M, \mathbb{F}, Y, h, v, \pi)$ is used, where $\mathbb{F}$ is a finite field and Y is a finite set, both sufficiently large. Again, we assume that h is a one-way hash function, $h : M \cup \mathbb{F} \to Y$. The trusted party TP is required only to keep the public keys and to prevent repeated signing. The start of the keys generation should be initiated in a suitable coordinated way.

Key generation and share distribution

Signer $S_i, i = 1, \ldots, n$

1. chooses randomly a non-zero element $x_i \in \mathbb{F}$ and makes (i, x_i) public
2. chooses randomly $s_j \in \mathbb{F}$ (secret key) for each $j = 1, \ldots, v$ such that $(j - 1) \bmod n + 1 = i$
3. computes the value $p_j = h(s_j)$ and sends (j, p_j) to TP
4. chooses randomly a polynomial $f_j(x) = f_{j,0} + f_{j,1}x + \cdots + f_{j,t-1}x^{t-1}$ satisfying $f_j(0) = f_{j,0} = s_j$
5. computes $s_{i',j} = f_j(x_{i'}), i' = 1, \ldots, n$
6. sends $s_{i',j}$ by a secure channel to the signer $S_{i'}, i' = 1, \ldots, n, i' \neq i$ (secret share for the signer $S_{i'}$)

Trusted Party TP

1. verifies that each p_j is from a proper S_i (a one-time signature of S_i can be used for signing the pair (j, p_j))
2. makes $(p_1, p_2, \ldots, p_v)$ public, registered to the name of the group.

Signing

Signer $S_i, i \in \{i_1, i_2, \ldots, i_t\}$

1. finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $q = h(m)$ and registers this value with TP; if q is different from a value already registered with TP, S_i stops signing
3. sends the partial signature $(s_{i,j_1}, s_{i,j_2}, \ldots, s_{i,j_r})$: the triple (i, j_k, s_{i,j_k}) is sent to S_{i_q} where $q = (j_k - 1) \bmod t + 1, k = 1, 2, \ldots, r$
4. using Lagrange interpolation, based on the conditions $f_{j_k}(x_{i_1}) = s_{i_1,j_k}$, $f_{j_k}(x_{i_2}) = s_{i_2,j_k}, \ldots, f_{j_k}(x_{i_t}) = s_{i_t,j_k}$, recovers the polynomial $f_{j_k}(x)$ for each complete t-tuple $((i_1, j_k, s_{i_1,j_k}), \ldots, (i_t, j_k, s_{i_1,j_k}))$ received.
5. for each polynomial f_j recovered, finds $s_j = f_{j,0}$
6. for each polynomial f_j recovered, sends (m, j, s_j) to V.

Verification

Verifier V

1. after receiving the first triple (m, j_k, s_{j_k}) finds $(j_1, j_2, \ldots, j_r) = \pi(m)$
2. computes $q' = h(m)$
3. fetches p_{j_k}, $k = 1, \ldots, r$ and q from TP
4. waits until all triples (m, j_k, s_{j_k}), $k = 1, \ldots, r$ are received
5. checks whether $p_{j_k} = h(s_{j_k})$, $k = 1, \ldots, r$ and $q' = q$.

As in Part 5.2, each of the signers knows at most $\lceil v/n \rceil$ secret keys of the group. Therefore, $(t-1)$ signers will not be enough to sign a message only if $\lceil v/n \rceil (t-1) < r_0$, where r_0 is the minimum signature length for messages under consideration.

The computation of the shares involves the time for $\max(n, n \lceil v/n \rceil \approx v)$ evaluations of a polynomial of degree $t-1$ and the same number of applications of h, since the signers may compute in parallel. Similarly, only the time for $\lceil r/t \rceil$ Lagrange interpolations of a polynomial of degree $t-1$ is required. In addition, V and each partial signer must compute $\pi(m)$ and V must compute r values of the function h.

6.1 Special Case: $t = 1$

A particular case of interest in this scheme is when $t = 1$, which depicts the anycast model. The anycast authentication problem was discussed in [2] and a solution was proposed based on a conventional digital signature. Briefly, the anycast model represents the situation where any of a group of n servers (signers) may provide the same (equivalent) service to a client (verifier). The method of nominating the actual signer is an optimization problem, and it is done by the network infrastructure based on a number of criteria such as less communication overhead, more available memory, and others. In the solution, an additional party - a group coordinator - may behave as the original signer in our $(1, n)$-threshold scheme while the servers behave as the proxies. The original signer delegates his

power to n proxy signers and the verifier verifies a message from "one" of the proxy signers. Although the above $(1, n)$- threshold scheme of one-time signature is theoretically correct, it is not of practical concern since the signer needs to generate different secret keys for each proxy correspondence.

7 Conclusion and Future Work

In this paper, we have proposed several schemes related to authentication with one-time signature. The first case deals with the implementation of a one-time signature in proxy delegation; the second shows how to use a (t, n) threshold one-time signature in a group of signers, and the third scheme combines the above two ideas into a (t, n) threshold proxy one-time signature. An extension to this work, left as an open problem, is to design a one-time signature scheme to prevent multiple message signing using the same set of one-time signature keys.

References

1. M. Al-Ibrahim and J. Pieprzyk, "Authentication of transit flows and K-siblings one-time signature," in *Advanced Communications and Multimedia Security*, B. Jerman-Blazic and T. Klobucar, ed., pp. 41–55, Kluwer Academic Publisher, CMS'02, Portoroz – Slovenia, September 2002.
2. M. Al-Ibrahim and A. Cerny, "Authentication of anycast communication," in the proc. of Second MMM-ACNS'03, St-Petersburg, Russia, LNCS 2776, pp. 425–429, Springer-Verlag, 2003.
3. J.N.E. Bos, D.Chaum, "Provably unforgeable signatures," *Advances in Cryptology – CRYPTO '92*, Ernest F. Brickell ed., LNCS 740, pp.1–14, Springer-Verlag, 1992.
4. C. Boyd, "Digital Multisignatures," in *Cryptography and coding* (H. Beker and F. Piper, eds.), pp. 241–246, Clarendon Press, 1989.
5. Y. Desmet and Y. Frankel. "Threshold cryptosystems," in G. Brassard, editor, *Advances in Cryptology – Crypto'89*, LNCS 435, pp. 307–315, Springer-Verlag, 1990.
6. Y. Desmedt, "Society and group oriented cryptography: a new concept," in *Advances in Cryptology – Proceedings of CRYPTO '87*, C. Pomerance, ed., LNCS 293, pp. 120–127, Springer-Verlag, 1988.
7. T. ElGamal, "A Public key cryptosystem and a signature scheme based on discrete Logarithms, " *IEEE Trans. on Inform. Theory*, vol. IT-31, pp. 469–472, July 1985.
8. S. Even, O Goldreich, S. Micali. "On-line/Off-line digital signatures," *Journal of Cryptology*, volume 9, number 1, pp. 35–67, 1996.
9. R. Gennaro and P. Rohatchi, "How to sign digital streams," *Advances in Cryptology – CRYPTO'97*, LNCS 1249, pp. 180–197, Springer-Verlag, 1997.
10. O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, 33(4):pp. 792–807, October 1986.
11. S. Goldwasser, S. Micali, and C. Rackoff, "A Digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal on Computing, 17, pp. 281–308, 1988.

12. H. Kim, J. Baek, B. Lee, and K. Kim, "Secret computation with secrets for mobile agent using one-time proxy Signature," Proc. of *SCIS'2001*, pp. 845–850, IEEE press, 2001.
13. L. Lamport, "Constructing digital signatures from a one-way function," Technical report CSL-98, SRI International, Palo Alto, 1979.
14. C.M. Li, T. Hwang, and N.Y. Lee, "Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders," in A. De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, LNCS 950, pp. 194–204, Springer-Verlag, 1995.
15. M. Mambo, K. Usuda, E. Okamoto, "Proxy signatures for delegating signing operation," IEICE Trans. Fundamentals, vol. E79-A, no.9, pp.1338–1354, 1996.
16. A. Menezes, P. Van Oorschot, and S. Vanstone. "Handbook of Applied Cryptography," *CRC Press*, Boca Raton, 1997.
17. R. Merkle. "A Certified digital signature," *Advances in Cryptology – CRYPTO'89,* LNCS 435, pp. 218–238, Springer-Verlag, 1989.
18. A. Perrig, R. Canetti, J.D. Tygar, D. Song, "Efficient Authentication and signing of multicast streams over lossy channels," *IEEE Symposium on Security and Privacy*, pp. 56–73, 2000.
19. A. Perrig. "The BiBa one-time signature and broadcast authentication protocol," *ACM, CCS'01*, pp. 28–37, 2001.
20. J. Pieprzyk, T. Hordajano, and J. Seberry. "Fundamentals of computer security," Springer-Verlag, 2003.
21. J. Pieprzyk, H. Wang, C. Xing. "Multiple-time signature schemes secure against adaptive chosen message attack," *SAC'03* (10th workshop on Selected Areas of Cryptography), LNCS, Springer-Verlag, 2003, to appear.
22. M.O. Rabin, "Digitalized signatures," R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, Foundations of Secure Computation, pp. 155–168, Academic Press, 1978.
23. L. Reyzin and Natan Reyzin, "Better than BiBa: short one-time signatures with fast signing and verifying," *ACISP'02*, LNCS 2384, pp. 144–152, Springer-Verlag, 2002.
24. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.
25. P. Rohatchi. "A Compact and fast hybrid signature scheme for multicast packet authentication," in *Proc. of 6th ACM Conference on Computer and Communications Security*, pp. 93–100, 1999.
26. A. Shamir. "How to share a secret," *Communications of the ACM*, 22:612–613, 1979.
27. K. Zhang, "Threshold proxy signature scheme," In proc. of *ISW'97,* LNCS 1396, pp. 282–290, Springer-Verlag, 1997.

A Threshold GQ Signature Scheme*

Li-Shan Liu, Cheng-Kang Chu, and Wen-Guey Tzeng

Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 30050
{gis89566,ckchu,tzeng}@cis.nctu.edu.tw

Abstract. We proposed the first threshold GQ signature scheme. The scheme is unforgeable and robust against any adaptive adversary if the base GQ signature scheme is unforgeable under the chosen message attack and computing the discrete logarithm modulo a safe prime is hard. Furthermore, our scheme can achieve optimal resilience by some modification.

Keywords: threshold signature scheme, GQ signature scheme.

1 Introduction

A threshold cryptographic protocol involves a set of players together, who each possesses a secret share, to accomplish a cryptographic task via exchange of messages. Threshold cryptographic protocols provide strong security assurance and robustness against a number of malicious attackers under a threshold. For example, in a (t, n)-threshold signature scheme, as long as $t + 1$ servers agree, they can jointly produce a signature for a given message even some other servers intend to spoil such process. Also, as long as the adversary corrupts less than $t + 1$ servers, it cannot forge any valid signature.

The signature scheme proposed by Guillou and Quisquater [GQ88], called the GQ signature scheme here, are used in many cryptographic protocols, such as forward-secure signature scheme [IR01], identity-based signature scheme [DQ94], etc. To our best knowledge, there are no threshold versions of this important signature scheme in the open literature. Therefore, in this paper we study the threshold signature protocol based on the GQ signature scheme. Our scheme is secure in the adaptive adversary model and can achieve optimal resilience, that is, the adversary can corrupt up to a half of the players. We also extend our work to the forward signature paradigm in the complete version of this paper.

Related work. Threshold schemes can be generally applied by the secure multi-party computation, introduced by [Yao82,GMW87]. However, since these solutions based on the protocol that compute a single arithmetic or Boolean gate, the schemes are inefficient. The first general notion of *efficient* threshold

* Research supported in part by National Science Council grant 91-2213-009-101 and MOE Excellence grant 91-E-FA04-1-4, Taiwan, ROC.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 137–150, 2003.

cryptography was introduced by Desmedt [Des87]. It started many studies on threshold computation models and concrete threshold schemes based on *specific* cryptosystems such as DSS, RSA, etc.

For the DSS scheme, the first solution was proposed by Cerecedo et al. [CMI93] under a non-standard assumption. Then Gennaro et al. [GJKR96b] provided another solution with security relying only on the regular DSS signature scheme. Canetti et al. [CGJ+99] and Frankel et al. [FMY99a] improved the security against the *adaptive* adversary. Jarecki and Lysyanskaya [JL00] furthermore removed the need of reliable erasures from adaptively. Jarecki [Jar01] summarized these techniques.

On the other hand, threshold RSA problem is more interesting. Since the share holders do not know the order of the arithmetic group, the polynomial interpolation is not as trivial as those of discrete-log based threshold cryptosystems. Desmedt and Frankel [DF91] provided the first heuristic threshold RSA scheme without security analysis. Later, they extended their work with a security proof [FD92]. Santis et al. [SDFY94] also proposed another provably secure threshold RSA scheme. Both [FD92] and [SDFY94] tried to avoid the polynomial interpolation. However, these schemes are complicated and need either interaction or large share sizes. Besides, they do not consider the robustness property. The robust threshold RSA schemes were then proposed by Gennaro et al. [GJKR96a] and Frankel et al.[FGY96]. Subsequently, some more efficient and simpler schemes for threshold RSA in the static adversary model were presented [FGMY97][Rab98]. These schemes take an extra layer of secret sharing so that much interaction are needed. Shoup [Sho00] provided a much simpler scheme without any interaction in partial signature generation. For adaptively-secure threshold RSA, there exist some solutions by [CGJ+99][FMY99a][FMY99b] as well. These protocols developed many techniques for designing secure threshold protocols.

2 Preliminaries

Guillou and Quisquater [GQ88] proposed an identification scheme. Then the GQ signature scheme is obtained by the standard Fiat-Shamir transformation [FS86]. The security of GQ signature scheme is based on the assumption that computing e-th root modulo a composite is infeasible without knowing the factors. The scheme is as follows (security parameter: k_1, k_2).

1. *Key generation*: let $n = pq$ be a k_1-bit product of two safe primes and e be a (k_2+1)-bit random value. The private key of a player is (n, e, s), where $s \in_R Z_n^*$ and the corresponding public key is (n, e, v), where $v = 1/s^e \bmod n$. Note that the user need not know p and q and thus n can be used by all users.
2. *Signing*: let h be a publicly defined cryptographic strong hash function, such as SHA-1. Given a message M, the signer computes the signature (σ, z) as follows:

- Randomly select a number $r \in Z_n^*$ and compute $y = r^e \bmod n$ and $\sigma = h(y||M)$.
- Compute $z = r \cdot s^\sigma \bmod n$.

3. *Verification*: the verifier checks whether $h(M||z^e v^\sigma \bmod n) = \sigma$.

A *threshold signature scheme* consists of the following three components:

1. *Key generation*: there are two categories in generating the keys and distributing shares of them to the participated players. In the dealer model, a dealer chooses the keys and distributes the shares to the players. In the distributed key generation model, all players together compute their key shares together.
2. *Distributed signing*: there are two phases: partial signature generation and signature construction. In the partial signature generation phase, the players communicate with each other and each produces a partial signature for the given message M. Then, in the signature construction, any one who has a number of valid partial signatures over a threshold can compute a valid signature for M.
3. *Verification*: any one can verify the validity of a signature for a message given the public key.

The security of threshold signature scheme includes both unforgeability and robustness as defined below.

Definition 1 (Unforgeability). *A (t, l)-threshold signature scheme is unfogeable in certain adversarial model if, except a negligible probability, no adversary in that model corrupts up to t players can produce a valid signature on a message that was not signed by any uncorrupted player.*

Another important property of threshold schemes is *robustness*. It ensures that the protocol can output a correct result as long as the adversary controls at most t players.

Definition 2 (Robustness). *A (t, l)-threshold signature scheme is t-robust in certain adversarial model if even the adversary in that model controls up to t players, the signature scheme is guaranteed to complete successfully.*

A threshold signature scheme is called *t-secure* if the two above properties are satisfied.

Definition 3 (Security of threshold signature). *A (t, l)-threshold signature scheme is t-secure in certain adversarial model if it is both unforgeable and t-robust in that model.*

An *adaptive adversary* is a probabilistic polynomial time Turing machine which can corrupt players dynamically, that is, it can corrupt a player at any time during execution of the protocol. Nevertheless, the total number of players it can corrupt is under the threshold.

Two distribution ensembles $\{X_n\}$ and $\{Y_n\}$ are (computationally) *indistinguishable* if for any probabilistic polynomial-time distinguisher D and any polynomial $p(n)$, there is an integer n_0 such that for any $n \geq n_0$,

$$|\Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1]| < 1/p(n).$$

Input: security parameters k_1, k_2.
The dealer generates and distributes keys as follows:

1. Choose two $\lceil k_1/2 \rceil$-bit random primes p, q such that $p = 2p' + 1, q = 2q' + 1$, where p', q' are also primes. Let $n = pq, m = p'q'$, and g a generator of G_n.
2. Choose a random polynomial $f(x) = a_0 + a_1 x + \ldots + a_t x^t$ over Z_m of degree t. Let $s = g^{a_0} \bmod n$ be the main secret and hand $s_i = g^{f(i)} \bmod n$ to player P_i secretly.
3. Randomly choose a $(k_2 + 1)$-bit value e, such that $\gcd(e, \phi(n)) = 1$ and $\gcd(e, L^2) = 1$, where $L = l!$. Compute $v = 1/s^e \bmod n$.
4. Let $SK_i = (n, e, g, s_i)$ be the secret key of player P_i, and broadcast the public key $PK = (n, e, g, v)$.

Fig. 1. TH-GQ-KeyGen: Generating keys

The *discrete logarithm* over a safe prime problem is to solve $DLog_{\tilde{g}}\tilde{h} \bmod \tilde{p}$ from given $(\tilde{p}, \tilde{g}, \tilde{h})$, where $\tilde{p} = 2\tilde{p}' + 1$ is prime, $\tilde{p}'$ is also prime, and $\tilde{g}$ is a generator of the quadratic subgroup $G_{\tilde{p}'}$ of $Z^*_{\tilde{p}}$. We assume that no probabilistic polynomial-time Turing machine can solve a significant portion of the input. Let I_n be the (uniform) distribution of the size-n input. Then, for any probabilistic polynomial-time Turing A and polynomial $p(n)$, there is n_0 such that for any $n \geq n_0$,

$$\Pr_{\tilde{p},\tilde{g},\tilde{h} \in I_n} [A(\tilde{p}, \tilde{g}, \tilde{h}) = DLog_{\tilde{g}}\tilde{h} \bmod \tilde{p}] < 1/p(n).$$

3 Threshold GQ Signature Scheme

In our threshold GQ signature scheme, the dealer generates a public/secret key pair and distributes the shares of the secret key to the players. To sign a message distributively, each player produces a partial signature. If there are more than $t+1$ valid partial signatures, we can construct the signature of the message from the valid partial signatures.

3.1 Generating Keys

The *key generation* process is shown in Figure 1. Let $\{P_i\}$ be the set of l participating players and $L = l!$. There are two security parameters k_1 and k_2. The dealer chooses two safe primes $p = 2p' + 1$ and $q = 2q' + 1$, each of length $\lceil k_1/2 \rceil$ bits, where p' and q' are also primes. Let $n = pq$, $m = p'q'$, Q_n the set of all quadratic residues modulo n, and g a generator of Q_n. The order of Q_n is m. Hereafter, all group computations are done in Q_n and the corresponding exponent arithmetic is done in Z_m.

The dealer then chooses a random degree-t polynomial $f(x)$ over Z_m and gives the share $s_i = g^{f(i)}$ to the player P_i. Note that the share given to player P_i is $g^{f(i)}$, instead of $f(i)$. The shared secret key is thus $s = g^{f(0)}$. The dealer then

Signing message
Input: message M;

Distributed Signing:

1. All players perform INT-JOINT-EXP-RVSS to generate $y = g^{f_r(0)e} \bmod n$ such that each player P_i gets his share $f_r(i)$ and computes $r_i = g^{f_r(i)} \bmod n$.
2. All players compute $\sigma = H(y, M)$.
3. All players perform INT-JOINT-ZVSS such that each player P_i gets a share $f_c(i)$ and computes $c_i = g^{Lf_c(i)} \bmod n$. All secret information, except c_i, generated in this step is erased.
4. Each player P_i computes his partial signature $z_i = (r_i s_i^{\sigma})^L c_i \bmod n$.

Signature construction:

5. Compute

$$z' = \prod_{j=1}^{t+1} z_{i_j}^{\lambda_{i_j} L} \bmod n,$$

where $z_{i_1}, z_{i_2}, \ldots, z_{i_{t+1}}$ are $t+1$ valid signatures and λ_{i_j}'s are the corresponding interpolation coefficients.
6. Find integers a, b for $L^2 a + eb = 1$, and compute $z = z'^a \cdot (y/v^{\sigma})^b \bmod n$.
7. The signature of message M is (z, σ).

Verifying signature
Input: (PK, M, SIG) where $PK = (n, e, g, v), SIG = (z, \sigma)$.

1. Compute $y' = z^e v^{\sigma} \bmod n$.
2. Accept the signature if $\sigma = H(y', M)$.

Fig. 2. TH-GQ-Sig & TH-GQ-Ver: Signing and verifying message

chooses a random $(k_2 + 1)$-bit value e with $\gcd(e, \phi(n)) = 1$ and $\gcd(e, L^2) = 1$ and computes $v = 1/s^e \bmod n$. In summary, the public key PK of the scheme is (n, e, g, v) and the secret share of player P_i is $SK_i = (n, e, g, s_i)$.

3.2 Signing Messages

The message signing protocol consists of two phases: *distributed signing* and *signature construction*, shown in Figure 2.

Distributed signing. To sign a message M, players jointly compute $y = r^e \bmod n$ first, where r is a random secret value. However, for the simplicity of partial signature construction, we use $g^{f_r(x)}$ instead of $f_r(x)$ to share the value r. That is, each player P_i gets a share $r_i = g^{f_r(i)} \bmod n$ and r is defined

as $g^{f_r(0)}$. Therefore, we provide a protocol INT-JOINT-EXP-RVSS, shown in Figure 5, letting all players jointly compute $y = g^{f_r(0)\cdot e}$, and each player P_i get his own share $g^{f_r(i)}$. Thus $\sigma = H(y, M)$ can be easily computed by all players, where H is the chosen one-way hash function.

Then All players jointly execute INT-JOINT-ZVSS[1] protocol to share a zero-constant t-degree polynomial $f_c(x)$, and each player P_i holds a share $c_i = g^{Lf_c(i)} \bmod n$. This randomized polynomial is generated for the security proof, described in Section 3.4. All other secret information generated in INT-JOINT-ZVSS are then erased (the erasing technique [CFGN96,CGJ+99]). Finally, each player P_i computes his partial signature $z_i = (r_i s_i^{\sigma})^L c_i \bmod n$.

Signature construction. To compute the signature for M, we choose $t+1$ valid partial signatures $z_{i_1}, z_{i_2}, ..., z_{i_{t+1}}$ and compute the interpolation

$$\begin{aligned} z' &= \prod_{j=1}^{t+1} z_{i_j}^{\lambda_{i_j} L} \bmod n \\ &= (g^{\sum_{j=1}^{t+1} \lambda_{i_j}(f_r(i_j)+\sigma f(i_j)+f_c(i_j))})^{L^2} \bmod n \\ &= (rs^{\sigma})^{L^2} \bmod n, \end{aligned}$$

where λ_{i_j} is the jth interpolation coefficient for the set $\{i_1, i_2, \ldots, i_{t+1}\}$. Since $\lambda_{i_j} L$ is an integer, we can compute z' without knowing the factorization of n (and thus m). Moreover, because that $\gcd(L^2, e) = 1$, we can find integers a, b such that $L^2 a + eb = 1$ and compute the signature z for M as:

$$z = z'^a \cdot (y/v^{\sigma})^b = (rs^{\sigma})^{L^2 a} (rs^{\sigma})^{eb} = rs^{\sigma} \bmod n$$

Remark. Since the sharing polynomials $f(x), f_r(x)$, and $f_c(x)$ are over the exponents, the partial signature $z_i = (r_i s_i^{\sigma})^L c_i$ is a share of a degree-t polynomial in the exponent. Thus, we need only $t+1$ shares to interpolate z'. This helps us to modify the protocol to achieve optimal resilience. The detail is described in Section 3.5.

3.3 Verifying Signatures

The verification procedure is straightforward, as defined by the GQ signature scheme, shown in Figure 2.

3.4 Security Analysis

Our threshold GQ signature scheme is secure against the chosen message attack in the adaptive adversary model. That is, as long as the adaptive adversary

[1] INT-JOINT-ZVSS is just like INT-JOINT-RVSS in Figure 4, except that each player sets his secret a_{i0}, b_{i0} to be zero.

Input: message M and the corresponding signature (σ, z)
Let $\mathcal{B}$ be the set of corrupted players and $\mathcal{G}$ the set of honest players at that time. Randomly choose $s_i \in Q_n$ for P_i, $1 \le i \le l$.

1. Let $y = z^e v^\sigma \bmod n$.
2. Execute $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$ on input y.
3. Execute INT-JOINT-ZVSS and assign each corrupted player P_i a share $c_i = g^{f_c(i)L} \bmod n$.
4. Randomly select a set $\Lambda' \supseteq \mathcal{B}$ of t players and for each $P_j \notin \Lambda'$, compute

$$z_j^* = z^{\lambda_{j,0} L} \cdot \prod_{k \in \Lambda'} (r_k s_k^\sigma g^{f_c(k)})^{\lambda_{j,k} L} \bmod n,$$

set

$$c_j^* = z_j^* / (r_j s_j^\sigma)^L \bmod n,$$

and erase c_j.
5. Broadcast all partial signatures z_i^* for $i \in \mathcal{G}$. (Note that $z_i^* = z_i$ for $P_i \in \Lambda' - \mathcal{B}$.)

Fig. 3. $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$: Simulator of TH-GQ-Sig

controls less than $t+1$ players, it cannot forge a valid signature without interacting with un-corrupted players. The adversary cannot interrupt the un-corrupted players to cooperatively obtain a valid signature for a message, either.

We need a simulator $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$ to simulate the view of execution of the TH-GQ-Sig scheme producing a signature (σ, z) for a message M. The simulator is shown in Figure 3.

Lemma 1. *If the adaptive adversary corrupts at most t players, its view of an execution of TH-GQ-Sig on input message M and output signature (σ, z) is the same as the view of an execution of $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$ on input M and signature (σ, z).*

Proof. Assume that $\mathcal{B}$ is the set of corrupted players and $\mathcal{G}$ is the set of un-corrupted (honest) players up to now. In the beginning (the key generation stage), the simulator emulates the dealer to randomly assign $s_i \in Q_n$ to player P_i, $1 \le i \le l$. These s_i's remain fixed for many rounds of simulation of TH-GQ-Sig. Let $y = z^e v^\sigma \bmod n$, which is the correct $r^e \bmod n$. Since y is distributively computed by all players in TH-GQ-Sig, the simulator runs $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$ (Figure 6) on input y to simulate the execution of INT-JOINT-EXP-RVSS protocol. In Step 3, the simulator runs INT-JOIN-ZVSS on behalf of honest players and assigns each corrupted player P_i a share $c_i = g^{f_c(i)} \bmod n$, where $f_c(x)$ has a zero constant. Now, the corrupted players P_i get s_i, r_i and c_i. Their partial signatures $z_i = (r_i s_i^\sigma)^L c_i \bmod n$ need be fixed since the adversary corrupts them. Let $\Lambda' \supseteq \mathcal{B}$ be a set of t players. We fix the partial signatures of the players in Λ'. For un-corrupted players $P_j \notin \Lambda'$, we set their partial signatures to be compatible with those in Λ', that is, the shares of any $t+1$ players result in the same signature. This is done by setting their partial signatures as

$$z_j^* = z^{\lambda_{j,0}L} \cdot \prod_{k \in \Lambda'} (r_k s_k^\sigma g^{f_c(k)})^{\lambda_{j,k}L} \bmod n,$$

where $\lambda_{j,k}$'s are the interpolation coefficients for computing the jth share from the set of shares $\{0\} \cup \{k | P_k \in \Lambda'\}$. The simulator also sets the new shares $c_j^* = z_j^*/(r_j s_j^\sigma)^L$ to the players $P_j \notin \Lambda'$ and erases the old shares c_j. These c_j^* make the un-corrupted players have the consistent relation for r_j, s_j, c_j^* and z_j^*.

We see how the simulator produces an indistinguishable distribution for the adversary:

1. The simulator runs $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$ which generates y in the proper distribution.
2. The simulator performs INT-JOINT-ZVSS on behalf of honest players. This is the same as what TH-GQ-Sig does in Step 3. Thus, the distribution for c_i's is the same.
3. The partial signatures z_i's of all players are consistent since the shares of any $t+1$ players produces the right signature (σ, z) (by adjusting c_i's). Therefore, they have the right distribution.
4. The erasing technique (for c_i's) is employed. As long as the simulated distribution is indistinguishable for the adversary after it corrupts a new player, the entire distribution is indistinguishable for the adversary after it corrupts up to t players. There is no inconsistency problem between corruptions of players.
5. The shares c_j's are adjusted to c_j^*'s for the un-corrupted players (up to now) so that even the adversary corrupts it later, the partial signature z_j^* is consistent with the possible check of equation $z_j^* = (r_j s_j^\sigma)^L c_j^*$.

In conclusion, the simulator $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$ produces an indistinguishable distribution for the adversary, who corrupts up to t players in an adaptive way.

□

We now show that our threshold GQ signature scheme is secure against the adaptive adversary under the chosen message attack.

For unforgeability, let $\mathcal{O}_{sig}$ be the signing oracle that a forger (in the centralized version) queries for signatures of messages. When $\mathcal{O}_{sig}$ returns (σ, z) for a message M, the simulator, on input M and (σ, z), outputs a transcript with an indistinguishable distribution for the adaptive adversary (in the distributed version). Thus, the adversary, who engaged several executions of the TH-GQ-Sig protocol, cannot produce an additional valid signature without cooperation of un-corrupted players.

Theorem 1. *If the underlying GQ signature scheme is unforgeable under the adaptive chosen message attack, the threshold GQ signature scheme in Figures 1 and 2 is unforgeable against the adaptive adversary who corrupts up to t players.*

Proof. Assume that the adversary $\mathcal{A}$, who controls up to t players during execution of the TH-GQ-Sig scheme and thus obtains signatures for $M_1, M_2, \ldots$, produces a valid signature for M, $M \neq M_i$ for $i \geq 1$. We construct a forger $\mathcal{F}$ to forge a signature of the underlying GQ signature scheme for an un-queried message using the procedure $\mathcal{A}$ and the signing oracle $\mathcal{O}_{sig}$ for the underlying GQ signature scheme.

Let (n, e, g, v) be the public key of the underlying GQ signature scheme. This is used in the (simulated) threshold GQ signature scheme also. First, since $\mathcal{F}$ does not know the corresponding secret key, in the key generation stage $\mathcal{F}$ assigns each player P_i a random secret share $s_i \in Q_n$. Then, it simulates all players and the adversary $\mathcal{A}$. When the adversary $\mathcal{A}$ intend to execute TH-GQ-Sig to produce a valid signature for M_i, $\mathcal{F}$ queries $\mathcal{O}_{sig}$ to obtain a signature (σ_i, z_i) and runs the simulator $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$, on input M_i and (σ_i, z_i), to produce a transcript T_i with right distribution (by Lemma 1) for $\mathcal{A}$. Therefore, $\mathcal{F}$ simulates $\mathcal{A}$, on input of these transcripts T_i's, to produce a valid signature (σ, z) for a new message M, $M \neq M_i, i \geq 1$. Thus, the underlying GQ signature is not unforgeable under the chosen message attack, which is a contradiction.

□

Theorem 2. *If computing the discrete logarithm modulo a safe prime is hard, the TH-GQ-Sig scheme in Figure 2 is t-robust against the adaptive adversary.*

Proof. If there is an adversary $\mathcal{A}'$ who participates TH-GQ-Sig on the input messages that it selected such that the honest players fail to generate a valid signature for a given message, then we can construct an extractor $\mathcal{E}$ to solve the discrete-log problem modulo a safe prime. That is, on input $(\tilde{p}, \tilde{g}, \tilde{h})$, $\mathcal{E}$ can compute $\mathrm{DLog}_{\tilde{g}}\tilde{h} \bmod \tilde{p}$ as follows, where $\tilde{p} = 2\tilde{p}' + 1$, $\tilde{g}, \tilde{h}$ are generators of $G_{\tilde{p}'}$.

First, $\mathcal{E}$ lets the dealer generate the related keys as usual (Figure 1) except that the dealer chooses (1) $p = \tilde{p}$ (and thus $p' = \tilde{p}'$) (2) $g = \tilde{g}^2 \bmod n$. Without loss of generality, we assume that $g \not\equiv 1 (\bmod q)$. Since g is the generator of both $G_{p'}$ and $G_{q'}$, g is a generator of Q_n. For another generator $\tilde{h}$, $\mathcal{E}$ simulates h-generation protocol [Jar01] with $\mathcal{A}'$ and outputs $h = \tilde{h}$. Using the instance (g, h), $\mathcal{E}$ performs TH-GQ-Sig with $\mathcal{A}'$ on behalf of the honest players. Now, we show that if $\mathcal{A}'$ hinders the signing protocol from producing a valid signature, $\mathcal{E}$ can compute $\mathcal{D}{=}\mathrm{DLog}_g h \bmod n$, and then outputs $\mathrm{DLog}_{\tilde{g}}\tilde{h} = 2\mathcal{D} \bmod \tilde{p}$.

Let us consider where the protocol may fail to produce the valid signature. First, in executing the INT-JOINT-RVSS scheme (or INT-JOINT-ZVSS, see Figure 4), if a corrupted player P_i distributes his shares $(f_i(j), f_i'(j))$, $1 \leq j \leq n$, that pass the verification equation (1), but do not lie on a t-degree polynomial, the extractor $\mathcal{E}$ can solve the system of $t+2$ linearly independent equations of the form $c_0 + c_1 j + c_2 j^2 + \ldots + c_t j^t = f_i(j) + \mathcal{D} f_i'(j)$ with $t+2$ unknown variables $c_0, c_1, \ldots, c_t$ and $\mathcal{D}$. Then, the extractor outputs $\mathcal{D}$.

Another situation that $\mathcal{A}'$ may cheat is on the zero-knowledge proof in executing INT-JOINT-EXP-RVSS (Figure 5). If the corrupted player P_i broadcasts $(A_i^*, B_i^*) \neq (g^{a_{i0}e}, h^{b_{i0}e})$ in Step 2, $\mathcal{E}$ extracts $\mathcal{D} = \mathrm{DLog}_g h$ as follows. Assume that $A_i^* = g^{a'} \bmod n$ and $B_i^* = h^{b'} \bmod n$. After executing Steps 2a-2c, $\mathcal{E}$ gets $R_i = r_i + da'e$ and $R_i' = r_i' + db'e$. Then $\mathcal{E}$ rewinds $\mathcal{A}'$ to run Steps 2b-2c again. This gives $\mathcal{E}$ another two equations $R_i^* = r_i + d^* a'e$ and $R_i'^* = r_i' + d^* b'e$. As long as $d \neq d^*$ (the probability of equality is negligible), $\mathcal{E}$ can solve the equations and get the four unknown variables r_i, r_i', a', b'. Since $\mathcal{E}$ knows the value m, the extractor computes $\mathcal{D}$ from $a_{i0} + \mathcal{D} b_{i0} = a' + \mathcal{D} b' \bmod m$.

□

Input: (n, g, h), where n is the product of two large primes and g, h are generators of Z_n^*.

1. Each player P_i chooses two random polynomials of degree t:

$$f_i(x) = a_{i0} + a_{i1}x + \ldots + a_{it}x^t, \quad f_i'(x) = b_{i0} + b_{i1}x + \ldots + b_{it}x^t$$

   where
   a) $a_{i0} = L^2\hat{s}, \hat{s} \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$.
   b) $b_{i0} = L^2\hat{s}', \hat{s}' \in_R \{0, \ldots, n^2(\lfloor n/4 \rfloor - 1)\}$.
   c) $a_{ik}, b_{ik} \in_R \{0, L, 2L, \ldots, L^3n^2\}$, $k = 1, \ldots, t$.
2. Each player P_i broadcasts $C_{ik} = g^{a_{ik}}h^{b_{ik}} \bmod n$, $1 \le k \le t$, and hands $f_i(j), f_i'(j)$ to player P_j, $j \in \{1, \ldots, l\}$.
3. Each player P_j verifies his shares received from each other P_i by checking:

$$g^{f_i(j)}h^{f_i'(j)} \equiv \prod_{k=0}^{t} C_{ik}^{j^k} \bmod n \tag{1}$$

   If the check fails for an index i, P_j broadcasts a *complaint* message against P_i.
4. Each player P_i who received a complaint from player P_j broadcasts the corresponding shares $f_i(j), f_i'(j)$.
5. Each player marks as *disqualified* any player that
   – received more than t complaints, or
   – answered to a complaint with values that falsify Eq. 1.
6. Each player then builds a common set of non-disqualified players $QUAL$. The secret x is now defined as $x = \sum_{i \in QUAL} a_{i0}$, and each player P_i holds a share $x_i = \sum_{j \in QUAL} f_j(i)$.

Fig. 4. INT-JOINT-RVSS

3.5 Achieving Optimal Resilience

The protocols presented up to now have not yet achieved optimal resilience since in Step 5 of TH-GQ-Sig we need find $t + 1$ valid partial signatures. Also, Step 2(b) of INT-JOINT-EXP-RVSS need reconstruct the challenge d. With the simple majority vote, the threshold t is $n/3$ at most. For better efficiency, we can use the the Berlekamp-Welch algorithm [WB86] to reconstruct the value in $O(tn)$.

We describe how to modify them to achieve optimal resilience ($n \ge 2t + 1$). The main difference is that each player proves correctness of its partial signature when it is issued. For this proof, in the key generation stage the dealer directly shares $f(i)$ (instead of $g^{f(i)}$) to player P_i. In addition, the dealer shares another random polynomial $f'(x)$ and broadcasts the coefficient references of $f(x)$ and $f'(x)$ in the unconditionally-secure form, that is, $A_i = g^{f(i)L}h^{f'(i)L}$ where h is another generator of G_n. The share of P_i is still set to $s_i = g^{f(i)}$, and all other operations are the same. Now, each P_i has $f(i), f_r(i)$ and $f_c(i)$, and the corresponding public information exist. When signing a partial signature, each player presents a non-interactive zero-knowledge proof of knowledge of $f(i), f_r(i)$

and $f_c(i)$. Therefore, in signature construction one can verify the validity of partial signatures.

To resolve the case for Step 2(b) of INT-JOINT-EXP-RVSS, we use the non-interactive zero-knowledge proof technique similarly. We can also replace the INT-JOINT-RVSS of jointly generating the challenge d with a coin-flip protocol (e.g. in additive form).

References

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC '96)*, pages 639–648. ACM, 1996.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Proceedings of Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 98–115. Springer-Verlag, 1999.

[CMI93] Manuel Cerecedo, Tsutomu Matsumoto, and Hideki Imai. Efficient and secure multiparty generation of digital signatures based on discrete logarithms. *IEICE Trans. Fundamentals*, E76-A(4):532–545, 1993.

[Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *Proceedings of Advances in Cryptology - CRYPTO '87*, volume 293 of *LNCS*, pages 120–127. Springer-Verlag, 1987.

[DF91] Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures. In *Proceedings of Advances in Cryptology - CRYPTO '91*, volume 576 of *LNCS*, pages 457–469. Springer-Verlag, 1991.

[DQ94] Olivier Delos and Jean-Jacques Quisquater. An identity-based signature scheme with bounded life-span. In *Proceedings of Advances in Cryptology - CRYPTO '94*, volume 839 of *LNCS*, pages 83–94. Springer-Verlag, 1994.

[FD92] Yair Frankel and Yvo Desmedt. Parallel reliable threshold multisignature. Technical Report TR-92-04-02, Dept. of EE and CS, U. of Winsconsin, April 1992.

[FGMY97] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Optimal-resilience proactive public-key cryptosystems. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pages 384–393. IEEE, 1997.

[FGY96] Yair Frankel, Peter Gemmell, and Moti Yung. Witness-based cryptographic program checking and robust function sharing. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC '96)*, pages 499–508. ACM, 1996.

[FMY98] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Robust efficient distributed rsa-key generation. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC '98)*, pages 663–672. ACM, 1998.

[FMY99a] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Adaptively-secure distributed public-key systems. In *Proceedings of 7th Annual European Symposium on Algorithms (ESA '99)*, volume 1643 of *LNCS*, pages 4–27. Springer-Verlag, 1999.

[FMY99b] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Adaptively-secure optimal-resilience proactive rsa. In *Proceedings of Advances in Cryptology - ASIACRYPT '99*, volume 1716 of *LNCS*, pages 180–194. Springer-Verlag, 1999.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of Advances in Cryptology - CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1986.

[GJKR96a] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust and efficient sharing of rsa functions. In *Proceedings of Advances in Cryptology - CRYPTO '96*, volume 1109 of *LNCS*, pages 157–172. Springer-Verlag, 1996.

[GJKR96b] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In *Proceedings of Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 354–371. Springer-Verlag, 1996.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing (STOC '87)*, pages 218–229. ACM, 1987.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Proceedings of Advances in Cryptology - CRYPTO '88*, volume 403 of *LNCS*, pages 216–231. Springer-Verlag, 1988.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In *Proceedings of Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 332–354. Springer-Verlag, 2001.

[Jar01] Stanislaw Jarecki. *Efficient Threshold Cryptosystems*. PhD thesis, MIT, 2001.

[JL00] Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *Proceedings of Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer-Verlag, 2000.

[Rab98] Tal Rabin. A simplified approach to threshold and proactive rsa. In *Proceedings of Advances in Cryptology - CRYPTO '98*, volume 1462 of *LNCS*, pages 89–104. Springer-Verlag, 1998.

[SDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing (STOC '94)*, pages 522–533. ACM, 1994.

[Sho00] Victor Shoup. Practical threshold signatures. In *Proceedings of Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer-Verlag, 2000.

[WB86] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction of algebraic block codes. U.S. Patent No. 4,633,470, December 1986.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations. In *Proceedings of 23th Annual Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164. IEEE, 1982.

A INT-JOINT-RVSS Protocol

The INT-JOINT-RVSS protocol is similar to the JOINT-RVSS protocol, except that we use unconditionally-secure VSS over integers [FGMY97,FMY98, FMY99b] instead.

B INT-JOINT-EXP-RVSS Protocol

Our INT-JOINT-EXP-RVSS (Figure 5) is based on an adaptively-secure distributed key generation protocol [CGJ+99] except for the composite modulus and an additional constant exponent. In the protocol, players first jointly perform INT-JOINT-RVSS to share a random secret x. To compute $y = g^{xe} \bmod n$, each player P_i broadcasts $A_i = g^{a_{i0}e}, B_i = g^{b_{i0}e}$ where $\sum_{i \in QUAL} a_{i0} = x$, and proves the knowledge of $a_{i0}e, b_{i0}e$ by simultaneous proof technique [CGJ+99, Jar01].

We also provide a simulation for INT-JOINT-EXP-RVSS in Figure 6. On input $y = g^{xe} \bmod n$, the simulator constructs the same adversarial view as in the real protocol running.

Input: (n, e, g, h), where n is the product of two large primes and g, h are generators of Z_n^*.

1. Each player P_i performs INT-JOINT-RVSS and gets the following secret outputs:
   - Polynomials $f_i(x) = a_{i0} + a_{i1}x + \ldots + a_{it}x^t$, $f_i'(x) = b_{i0} + b_{i1}x + \ldots + b_{it}x^t$ he generated randomly.
   - The shares $f_j(i), f_j'(i)$ sent from player j, where $j = 1, \ldots, l$.
   - The share $f(i) = \sum_{j \in QUAL} f_j(i)$ of the secret x.

   The public outputs are $C_{ik} = g^{a_{ik}} h^{b_{ik}} \bmod n$ where $i = 1, \ldots, l$, $k = 0, \ldots, t$, and the set $QUAL$ of non-disqualified players.
2. Each player $P_i, i \in QUAL$ broadcasts $A_i = g^{a_{i0}e} \bmod n$ and $B_i = h^{b_{i0}e} \bmod n$ such that $A_i B_i = C_{i0}^e$, and prove the knowledge of $a_{i0}e, b_{i0}e$:
   a) P_i chooses $r_i, r_i' \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and broadcasts $T_i = g^{r_i} \bmod n$, $T_i' = h^{r_i'} \bmod n$.
   b) All players jointly execute INT-JOINT-RVSS and then publicly reconstruct the random secret d.
   c) Each player P_i broadcasts $R_i = r_i + d \cdot a_{i0}e$ and $R_i' = r_i' + d \cdot b_{i0}e$.
   d) Each player P_j checks that $g^{R_i} \equiv T_i \cdot A_i^d \bmod n$ and $h^{R_i'} \equiv T_i' \cdot B_i^d \bmod n$ for $i = 1, \ldots, l$. If the check fails for some index i, P_j complains against P_i.
3. For each player P_i receives more than t complaints, P_j broadcasts $f_i(j)$. All players reconstruct a_{i0} and compute $A_i = g^{a_{i0}e} \bmod n$.
4. All players then compute $y = \prod_{j \in QUAL} A_j \bmod n$

Fig. 5. INT-JOINT-EXP-RVSS

Input: the result value y and parameters (n, e, g, h)

1. Perform INT-JOINT-RVSS on behalf of honest players.
2. Choose an uncorrupted player P_u, and do the following computation:
   - Compute $A_i = g^{a_{i0}e} \bmod n$ and $B_i = h^{b_{i0}e} \bmod n$ for $i \in QUAL \backslash \{u\}$.
   - Set $A_u^* = y \cdot \prod_{i \in QUAL \backslash \{u\}} A_i^{-1} \bmod n$ and $B_u^* = C_{u0}^e / A_u^* \bmod n$.
   - Choose $d^*, R_u, R'_u \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and set $T_u^* = g^{R_u} \cdot (A_u^*)^{-d^*} \bmod n$, $T_u'^* = g^{R'_u} \cdot (B_u^*)^{-d^*} \bmod n$
3. Broadcast A_i for player P_i,$i \in QUAL \backslash \{u\}$ and A_u^* for player P_u.
4. Perform Step 2a in the protocol on behalf of each player P_i,$i \in QUAL \backslash \{u\}$, and broadcast T_u^* and $T_u'^*$ for player P_u.
5. Perform INT-JOINT-RVSS on behalf of honest players as Step 2b in the protocol, and each P_i gets a share d_i. Pick a random t-degree polynomial $f_d^*(x)$ over integers such that $f_d^*(0) = d^*, f_d^*(i) = d_i$ for $i \in \mathcal{B}$. Erase all other secret information generated in INT-JOINT-RVSS.
6. Broadcast $f_d^*(i)$ for each honest player P_i.
7. Broadcast R_i, R'_i computed as Step 2c for player P_i, $i \in \mathcal{G} \backslash \{u\}$. Broadcast $R_u^*, R_u'^*$ for player P_u.
8. Verify the proof as in the protocol. If players in $\mathcal{B}$ receive more than t complaints, all other players reconstruct their secrets.
9. Erase all secret information except $f(i)$.

Fig. 6. $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$

Generalized Key-Evolving Signature Schemes or How to Foil an Armed Adversary

Gene Itkis and Peng Xie

Boston University Computer Science Dept.
111 Cummington St.
Boston, MA 02215, USA
{itkis, xp}@cs.bu.edu

Abstract. Key exposures, known or inconspicuous, are a real security threat. Recovery mechanisms from such exposures are required. For digital signatures such a recovery should ideally —and when possible— include invalidation of the signatures issued with the compromised keys. We present new signature schemes with such recovery capabilities.

We consider two models for key exposures: full and partial reveal. In the first, a key exposure reveals *all* the secrets currently existing in the system. This model is suitable for the pessimistic inconspicuous exposures scenario. The partial reveal model permits the signer to conceal some information under exposure: e.g., under coercive exposures the signer is able to reveal a "fake" secret key.

We propose a definition of *generalized key-evolving signature scheme*, which unifies forward-security and security against the coercive and inconspicuous key exposures (previously considered separately [5,18,11]). The new models help us address repudiation problems inherent in the monotone signatures [18], and achieve performance improvements.

Keywords: *digital signatures, forward-security, monotone signatures, key-evolving signature schemes, key exposures, coercion, recovery.*

1 Introduction

Secret key exposures are a well-known threat for cryptographic tools. Such exposures may be inconspicuous (e.g., undetected theft) or obvious (e.g., extortion). In either case, recovery from such compromises presents an challenge which must be addressed. Typically, such recovery is achieved by revocation of the compromised keys and re-issue of the new version of the keys to the attacked user. Both of these operations are very expensive and complicated. Alternative recovery methods are thus highly desirable.

Coercive key exposures have been considered previously; in particular, [18] proposed a mechanism for invalidating all the signatures generated with the extorted keys, but not any of the signatures issued by the legitimate signer both *before or after* the coercion attack (thus eliminating the need for revocation

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 151–168, 2003.

and re-issue).[1] The detection of inconspicuous key exposures was addressed previously in [11], which also suggested an alternative to revocation. A potential connection to the monotone signatures was noted in that paper as well; here we develop this connection further.

In this paper we consider both the coercive and inconspicuous key exposures. First, we focus on the coercive exposures: we discuss the original definition of the monotone signatures from [18], and highlight some of its shortcomings and potential pitfalls. Then we propose a new definition, which avoids these pitfalls. Our definition generalizes the monotone signatures of [18] on the one hand, and the key-evolving signatures [5][2] on the other, and as such, it may be of independent value. Our definition also allows us to address both the coercive/overt and inconspicuous exposures within one model. We then propose some schemes, which not only provide an improved functionality, but are also more efficient than the original monotone signatures. Finally, we prove some (tight) lower-bounds.

1.1 Monotone Signatures — Evolution of the Public Keys

Let us briefly review the monotone signatures as they were originally defined in [18]. The reader is encouraged to consult the original paper for the formal definitions.

The monotone signatures are motivated by a scenario when the signer is forced to reveal a key which would enable the attacker to generate valid signatures. How can the system security be recovered after the extortion attack is over?

Public Key evolution. It appears that whatever recovery mechanism is used, the public key must change. It therefore makes sense to consider explicitly a model with the evolving public keys. This is exactly what the monotone signatures attempt (without presenting it in exactly these terms).

Monotone signatures. The original definition of the monotone signatures allowed the verification algorithm (i.e., the public key) to be updated, e.g., after an attack. Namely, the signer under duress can reveal some (but not all!) of his secrets to the attacker. These extorted secrets enable the attacker to generate signatures valid under the *current* public key. However, after the signer is no longer under duress, the public key can be updated, so that all the signatures generated by the legitimate signer (both, before and after the update or the attack) remain valid, but all the signatures generated by the attacker (using

[1] Alternative methods of creation of a "subliminal channel" allowing the coerced signer to covertly inform authorities of the coercion were also proposed (see, e.g., Funkspiel schemes of [10]). In this case, the key may not be actually exposed and one may hope that the notification of the authorities for the extorted signatures eliminates the need of the expensive global recovery.

[2] Key-evolving signature schemes were introduced as a functional definition for forward-secure signatures [3,5,16,2,13,17,12,15]. They also served as a basis for new notions such as key-insulated [6], intrusion-resilient [14,12], and tamper-evident [11] signatures.

the extorted keys) are no longer valid under the *updated* verification algorithm. Thus, monotone signatures can be viewed as a generalization of the signature schemes which includes the revocation and re-issue.

Shortcomings of the original definition. Unfortunately, the definition of [18] gives the signer too much power: it undermines the non-repudiation property of the signatures — often their main functionality. Indeed, a signer can choose to sign any message "as an attacker" in the above scenario. Then, at any time later on, he may choose to update his public key so as to invalidate that signature.

1.2 Generalized Key-Evolving Signatures Schemes and Extortion-Resilience

In this paper, we modify the definition of the monotone signature schemes to limit this repudiation power of the signer. We also generalize the definition to include all the above mentioned schemes as its special cases. In particular we refine the secret key exposure model to include both undetected inconspicuous exposures as well as the extortions.

Clearing period: restricting repudiation. In particular, in our scenarios, similarly to the real life check payments, each signature — even if verified as valid — is not "cleared" for a pre-defined period of time. We limit the repudiation ability of the signers to that clearing period.

Similarly to the forward-secure signatures, in our system the legitimate signers obeying the protocol do not have the information that could allow them to back-date their signatures. Thus, an attacker can use the extorted secrets only to generate signatures dated after the extortion.

We use techniques similar to tamper-evidence [11] to invalidate the signatures generated by the extorted keys, with respect to the public keys updated by the signer after he is released.

Forcing signer communication and subliminal channels. Given the above, the attacker may try to prevent a public key update altogether or at least to make sure that the updated public key does not invalidate the signatures based on the extorted secrets. To achieve this the attacker may resort to holding the signer hostage until the extorted signatures are cleared, or even killing the signer after the extortion. We need to eliminate the motivation for the attacker to resort to such measures.

To achieve this we require the signer to contact the authority at least once during the clearing period and perform the public key update. Thus, even if the attacker holds the signer captive during the clearing period, the attacker still needs to allow the signer to contact the authority, in order to have the signatures with the extorted keys cleared. However, this communication between the signer and authority, unlike the communication with the untrusted verifiers, can utilize subliminal channels (e.g., similar to the Funkspiel [10]). Thus at the very least the authority would know that the particular keys had been extorted. And even if these keys must be treated as valid in order to protect the signer, the authority can take whatever steps are available. And in particular, it may be able to reset

the system afterwards so that the extorted keys become invalid even after the clearing period has passed. They would also, of course, be able to identify the extorted signatures potentially helping with arresting the attacker.

Thus, the attacker is faced with the choice of having her extorted signatures invalidated before they are cleared, or risking authority notification and subsequent tracing. It is our hope that since both of the options involve significant risks for the attacker, our scheme provides a significant deterrent for an attack.

2 Definitions

2.1 Functional Definitions

General Key-Evolving Signature Schemes (*GKS*). Intuitively, in *GKS* scheme, both the secret key and public key can evolve. The signer can invalidate extorted signature by updating the public key. Secret key evolution enable forward-security, which in turn enables the clearing period functionality.

We start with the functional definition of *GKS*: A *general key-evolving signature scheme GKS*=(**KeyGen**,**SUpd**,**PUpd**,**Sign**,**Ver**) is a collection of five (randomized) polynomial algorithms, where:

KeyGen is the *key generation* algorithm,
Input: a security parameter $k \in \mathsf{N}$ (given in unary), and the total number of periods, T,
Output: a pair $(\mathtt{SK}_0, \mathtt{PK}_0)$, the initial secret key and public key;

SUpd is the *secret key update* algorithm,
Input: the secret key $\mathtt{SK}_t$ for the current period $t < T$, and control message $\mathtt{c}$, which determines the update type: $\mathtt{c} \in \{\mathtt{sk\_only}, \mathtt{sk\&pk}, \mathtt{pk\_only}\}$, corresponding to updating only the secret key, both secret and public, and public key only[3], respectively,
Output: the new secret key $\mathtt{SK}_{t+1}$ and update message μ_t;

PUpd is the *public key update* algorithm,
Input: the current public key $\mathtt{PK}_t$ and the update message μ_t,[4]
Output: The new public key PK_{t+1};

Sign is the *signing* algorithm,
Input: the secret key $\mathtt{SK}_t$ for the current period t and the message M to be signed,
Output: signature sig_t of M (the time period t of the signature generation is included in sig_t);

[3] The `pk_only` option is included here for generality. In this paper we will not support this option: it is convenient to record the number of public key updates in the secret key; moreover, an update message for the public key update must be generated, and this typically requires information from the secret key. E.g., for the monotone signature schemes of [18] our **SUpd** algorithm's main function is to generate the update message.

[4] The update type `c` here can be inferred from the update message μ, if desired.

Ver is the *verification* algorithm,
Input: the public key $\mathtt{PK}_t$, a message M, and an alleged signature sig,
Output: **valid** or **fail**.

Intuitively, we count time as the number of key updates, see Experiment Forge in subsection 2.2 for the more precise definition. The key update frequency is selected by the user: it could correspond to physical time intervals (e.g., one update per day), or performed arbitrarily (e.g., more frequently when the signer feels vulnerable), or activity related (e.g., one update per each signature).

For simplicity we assume full synchronization of the system: namely, we assume that there are no outdated public keys. In particular, a signature generated at time t should never be verified using a public key $\mathtt{PK}_i$ from an earlier period $i < t$.

Completeness, Monotonicity, Non-Repudiation, Clearing Period. We require the *completeness* property (as in [18]). Namely, all signatures generated by a legitimate signer must be valid for *all* the subsequent legitimate public keys: $\mathbf{Ver}(\mathtt{PK}_i, M, \mathbf{Sign}(\mathtt{SK}_t, M)) = \mathbf{valid}$ for any message M and any time periods $i \geq t$. In particular, validity of a legitimately generated signature should not be changed by updates.

We also require *monotonicity* of the signatures: an invalid signature cannot become valid at a later time. Formally: $\mathbf{Ver}(\mathtt{PK}_{t'}, M, sig_t) = \mathbf{valid} \Rightarrow \mathbf{Ver}(\mathtt{PK}_j, M, sig_t) = \mathbf{valid}$ for all $j : t \leq j \leq t'$. [5]

The monotone signatures, by their very design, are intended to allow the signer to output alleged secret keys (and thus signatures) which look perfectly legitimate at the time they are output, but can be invalidated at a later time by an appropriate public key update. If unrestricted, this power builds in a repudiation mechanism contradicting the very design of the signatures.

As mentioned in the Introduction, we limit this repudiation power to the *clearing period* δ: if the signature remains valid after $\geq \delta$ public key updates, then it can never be repudiated by any subsequent updates. Namely, suppose that signature sig_i was generated at time period i, and let j be a time period at least δ public key updates after i; then we require that $\mathbf{Ver}(\mathtt{PK}_j, M, sig_i) = \mathbf{valid} \Rightarrow \mathbf{Ver}(\mathtt{PK}_{j'}, M, sig_i) = \mathbf{valid}$ for all $j' > j$, and thus by monotonicity for all $j' \geq i$.

Forward-Secure and Monotone Signatures. The above general key-evolving scheme definition includes as special cases the functional definitions for forward-secure and monotone signatures: forward-secure signatures [5] update only the secret keys, while the monotone signatures [18] update only the public keys (**SUpd** is used only to update the time period and generate the update message).

Key Exposures. In order to formalize key exposures we introduce a special function:

[5] This notion of monotonicity is slightly different from that of [18]: we do not require a signature to be valid for the "outdated" public keys preceding the signature — in fact, we rule out such a verification altogether.

Reveal the (possibly randomized) *secret key revealing* algorithm;
Input: the current secret key $\mathtt{SK}_t$, and number r of the previous known attacks (i.e., the number of times **Reveal** was used previously with the signer's knowledge);[6]
Output: alleged secret key $\mathtt{SK}'_t$: $\mathbf{Ver}(\mathtt{PK}_t, M, \mathbf{Sign}(\mathtt{SK}'_t, M)) = \mathbf{valid}$ for all messages M.

Intuitively, **Reveal** outputs key $\mathtt{SK}'$ given to the attacker when she tries to expose the signer's key $\mathtt{SK}$. For all exposure models below, $\mathtt{SK}$ and $\mathtt{SK}'$ should be indistinguishable at the exposure time. In particular, $\mathtt{SK}'$ should generate signatures valid for the current public key. But after the subsequent public key updates, $\mathtt{SK}$ and $\mathtt{SK}'$ are easily distinguished: $\mathtt{SK}'$ will now generate invalid signatures.

Three models of exposures could be considered: *full, partial* and *creative*. The first model corresponds to the attacker learning *all* the secrets of the signer, $\mathbf{Reveal}(\mathtt{SK}, r) = \mathtt{SK}$.

Partial reveal allows the signer to conceal some of his secrets from the attacker: i.e., the attacker obtains a subset of all the secrets of the signer. It is important to note that the set of the exposed secrets in this model is determined by the signer, and not by the attacker. This is the model used in [18].

Finally, the creative reveal model appears to give the signer the greatest defensive powers: the signer is allowed to "lie creatively" and generate the alleged secret key $\mathtt{SK}'$ in an arbitrary way. However, all the alleged secret keys $\mathtt{SK}'$ can be pre-computed and stored in the $\mathtt{SK}$ to be revealed at the appropriate times. Thus, the creative reveal is actually equivalent to the partial reveal model. So, in the rest of the paper we consider only the full and partial reveal models.

For the sake of simplicity, in this version of the paper, we consider partial and full models separately. However, a hybrid model allowing a combination of the reveal attacks is of interest as well. In particular, the full reveal is well-suited for the inconspicuous attacks, while the partial model can address the extortion attacks. Since in real life both types of attacks are possible, it would be useful to have schemes (and model) to handle both at optimal costs.

2.2 Security Definitions

GKS SECURITY. In *GKS*, time is divided into time periods. The time is incremented by each key update. The key update can be either secret key only update, or both secret key and public key update (as noted in the footnote 3, while theoretically it is possible to allow a "public key only" update, we do not support it in this paper). We use $P(t)$ to denote the number of the public key updates (i.e., $\mathtt{sk}\&\mathtt{pk}$'s) that occurred since the key generation and up to the current time period t. We allow the adversary F to adoptively obtain signatures and secret keys (using **Reveal** function). We model these two types of information

[6] The number of r of previous attacks is needed only for partial and creative reveal models discussed below. For the full reveal model, this number can be ignored or even unknown.

with two oracles: *Osig* and *Orvl*. The most natural assumption is to allow only the queries relating to the current time t. We expand the adversary powers to allow her to query signatures for the past (but not the future, since the future signatures may depend on the specific evolution path the system takes).

Specifically, given message M and time period $i \le t$, oracle $Osig_t(M, i)$ at time t returns the signature that the signer would have generated for the message M during the period i. In contrast, oracle $Orvl_t^{(\rho)}$ can be queried only about the present: when queried at time t, it returns the (alleged) secret key $\mathbf{Reveal}^{(\rho)}(\mathtt{SK}_t, r)$, for the appropriate number r of the previous attacks and the reveal type $\rho \in \{pr, fr\}$: partial or full reveal.[7]

We use the following experiments to define the security of *GKS*.

```
Experiment Forge_GKS(F, k, T, δ)
t ← 0; μ_t ← empty string;
(SK_t, PK_t) ← KeyGen(1^k, T)
repeat
    q ← F^{Osig_t, Orvl_t}(PK_t)
    if (q = sk_only) then SK_{t+1} ← SUpd(SK_t, sk_only)
    if (q = sk&pk) then
        (SK_{t+1}, μ_t) ← SUpd(SK_t, sk&pk)
        PK_{t+1} ← PUpd(PK_t, μ_t)
    t ← t + 1
until (q = forge)
(M, σ_i, j ≥ i) ← F
if Ver(M, σ_i, PK_j) = valid, i ≤ j ≤ T, and neither Osig(M, i) nor Orvl^{(fr)}(i)
   were queried, and either P(j) ≥ P(i) + δ or Orvl^{(fr)}(i') was not queried for
   any i' < i : P(i') = P(i)
   then return 1.
```

Definition 1 (*GKS* security).
Let $\mathcal{S} = (\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{SUpd}, \mathbf{PUpd}, \mathbf{Ver})$ *be a GKS-scheme with parameters* k, T, δ *as above, and let adversary* F *be any PPT algorithm. Say that* $\mathcal{S}$ *is* secure *if* $\mathsf{Prob}[Forge_{\mathcal{S}}(F, k, T, \delta) = 1]$ *is negligible (e.g.,* $\le \frac{1}{2^k}$*).*

Consider an attacker who broke in during time period t. Obviously, she can now generate valid signatures for that period at least until the public key update. Moreover, in the case of the full reveal, she can generate signatures for the

[7] In the case of the full reveal model, we can allow the adversary to query the past: the $Orvl_t(i)$ oracle can return $\mathtt{SK}_i$ for any given time period $i \le t$. Moreover, the number r can be omitted, since it does not affect the **Reveal** function. In contrast, for partial reveal, exposing past keys is problematic because the information revealed is likely to depend on the number of r of prior exposures, and exposing past keys may change that. Also, in practice, partial reveals are typically to be used to model known attacks (extortions), and each such reveal is likely to be followed by a sk&pk update. The number of such updates can then be recorded in the SK and used instead of r. Thus we keep r only for the notational convenience.

time period t which will remain valid even after the public key updates. So, intuitively what *GKS* security requires is that these be the only forgeries that the attacker can generate successfully. In other words, *GKS* security confines the damage to the period of the exposure. In particular, the attacker still cannot forge valid signatures for any of the time periods before or after t. In the case of partial reveal, the attacker cannot even generate signatures for period t which will remain valid after the subsequent public key updates.

CMA, Forward-Security and Monotone Signatures Security. If no updates are used in the experiment *Forge* (i.e., $q = forge$ is immediately selected by adversary using only the *Osig* oracle), then the above definition is equivalent to the standard definition of security against *adaptive chosen message attacks* [8].

If no public key updates are performed (i.e., only `sk_only` updates are allowed), then the above definition converges to that of *forward-security* [5].

The above definition also captures the *monotone signatures* of [18] as mentioned above: Allow only `sk&pk` updates and restrict the secret key update to change only the time period — thus, main purpose of **SUpd** is to produce the update message μ (assume $\delta = 1$ for the above definition, but the non-repudiation after the clearing period is not provided by [18]). Moreover, for monotone signatures, $\mathtt{SK} = \langle s_1, \ldots, s_T \rangle$ and $\mathbf{Reveal}(\mathtt{SK}, i) = \langle s_1, \ldots, s_i \rangle$.

Soundness. For monotone signatures described as above, the soundness defined by [18] means that without $Orvl(\geq i)$ any adversary has only a negligible probability of successfully forging a signature for a public key $\mathtt{PK}_i$. Our definition includes this notion of soundness and further refines it to allow `sk_only` updates and full reveal.

In principle, it may be interesting to consider *GKS* schemes with $\delta > 1$ or even variable (e.g. depending on time). Specifically, such schemes might offer better performance. But for the sake of simplicity, in the rest of this paper we assume $\delta = 1$.

3 Using *GKS* Signatures

This section contains informal discussion about how the signer may wish to use the *GKS* signatures, a sort of short "user's manual" draft. As discussed in the introduction, we consider two types of exposures: known (e.g., extortion) and inconspicuous (undetected theft). Clearly, a full reveal exposure, whether known or undetected, leads to a compromise of all the signatures issued during the period of the exposure — it is impossible to distinguish signatures generated with the same secret key. Dealing with these compromised signatures is one obvious challenge. Another challenge is presented by the requirement that if a signature remains valid for a certain amount of time (the clearing period), then it can never become invalid. This later requirement, in particular, implies that it must be impossible to "back-date" signatures, even after the exposures. We address this clearing period requirement by including the forward-security in our definitions and constructions. Using the forward-security, we can approach the first challenge also (though, forward-security by itself may not be enough to

address it completely): in the extreme case, the signer may wish to perform an update (which includes erasing of the previous secret key) immediately before and immediately after each signature. Then the exposed secrets are the least likely to compromise any legitimate signatures.

While it may be too expensive to follow the above tactics all the time, it is recommended to increase frequency of updates whenever the signer feels there may be an increased chance of a key exposure. In particular, it is recommended to perform an update whenever the user enters a more risk-prone situation, when an attack is more likely. Namely, when facing danger, the signer would trigger the **SUpd** algorithm to erase the secret key and generate the new `SK`, which may be revealed to the attacker. The attacker can use the revealed secrets to generate forged signatures. Such signatures must be valid under the current public key. However, as a matter of policy, they might not be honored until the clearing period has passed. The same policy should require that passing of the clearing period requires public key update to be performed subsequently to the signature in question. However, all the signatures generated using extorted secrets would have to be invalidated with the subsequent public key updates.

In general, it is important to consider explicitly all the parties involved: *signer*, *verifiers*, *attacker*, and *authority*. Intuitively, the last party—authority—is responsible for communicating with the signer and performing the public key updates in the verifiers. It is the responsibility of the authority to ensure that there are no spurious public key updates even when the signer's keys are exposed. As expected, the signer generates the signatures and the verifiers verify the signatures. The signer also maintains the secret key by performing secret key update **SUpd** at the end of each time period, which generates update message μ. The signer then communicates μ to the authority, which performs the public key update **PUpd** and distributes the new public keys to the verifiers. We assume that the verifiers leak the public keys to the attacker.

The introduction of the authority in order to receive the update message from the signer allows us to "cheat" by going outside the scheme. In particular, it allows the use additional (unexposed) secrets, used only for communication between the signer and the authority. This secret cannot be "tested": for example, the signer may have two passwords, one of which would be used only in the case of an attack to communicate to the authority that the signer is being held by an attack and his secret is extorted — in this case, the authority may emulate the normal update behavior, but would notify the police or take other appropriate measures. More sophisticated subliminal channels (e.g. [10]) could be deployed for this purpose to allow the signer to communicate more information to the authority.

4 Constructions

4.1 Construction for Full Reveal Model

GKS and tamper-evident signatures. A *GKS* scheme could be obtained by using tamper-evident signatures [11]. Tamper-evident signatures are key-evolving signature schemes, using only secret key evolution. Their main fea-

ture is inclusion of a special predicate Div: given two signatures, Div determines whether only one of them is generated by the legitimate user (and the other by the forger), provided that both signatures were generated after the latest (undetected) key exposure. The *GKS* security could be obtained by inclusion the latest legitimately generated tamper-evident signature into the public key. The verification procedure would then include the Div test between this signature and the signature being verified. The scheme below can be viewed as an optimization of the above approach. Perhaps the most surprising aspect of this scheme is that, despite its simple approach, it turns out to be optimal (as shown in Sec. 5). *GKS* CONCATENATION SCHEME. We propose a *GKS* scheme based on an arbitrary ordinary signature scheme. The idea behind this construction is quite simple: For each time period, a different ordinary secret-public key pair is used; these public keys are then published in the *GKS* public key. This leaves open the question of validity of the signatures generated after the latest public key update. For this, in each public key update we include a separate ordinary public key, used to certify the signing keys after this public key update and until the next one. After the next public key update, this certification key can be discarded, since the public keys it certified are now included directly in the *GKS* public key. Next we give the formal description of the scheme.

Let Σ be any ordinary signature scheme. We use instances of Σ for message signing and certification: S_t are used during time periods t, and $\mathtt{C}$ is used to certify $S_t.\mathtt{PK}$ for the current time periods.[8] $S_t.\mathtt{SK}, S_t.\mathtt{PK}$ denote the secret and public keys of S_t, generated by the $\Sigma.\mathbf{KeyGen}$ algorithm. We write $S_t.\mathbf{Sign}(m)$ to denote $\Sigma.\mathbf{Sign}(S_t.\mathtt{SK}, m)$ (similarly, $S_t.\mathbf{Ver}(m, sig) \stackrel{def}{=} \Sigma.\mathbf{Ver}(S_i.\mathtt{PK}, m, sig)$). Let $\vec{\mathtt{PK}}_{[i,j]} \stackrel{def}{=} S_i.\mathtt{PK}||\ldots||S_j.\mathtt{PK}$; $\vec{\mathtt{PK}}_j \stackrel{def}{=} \vec{\mathtt{PK}}_{[1,j]}$; $\vec{\mathtt{PK}}_0 \stackrel{def}{=} \oslash$ (i.e., the *empty string*). Intuitively, μ' below is a "current draft" update message, stored in $\mathcal{S}.\mathtt{SK}$. Let $\mathcal{S} = (\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{SUpd}, \mathbf{PUpd}, \mathbf{Ver})$.

$\mathcal{S}.\mathbf{KeyGen}$:
  $(\mathtt{C.SK}, \mathtt{C.PK}) \leftarrow \Sigma.\mathbf{KeyGen}(1^k)$;
  `return` $(\mathcal{S}.\mathtt{PK}_0 \stackrel{def}{=} \langle \vec{\mathtt{PK}}_0, \mathtt{C.PK}\rangle,\ \mathcal{S}.\mathtt{SK}_0 \stackrel{def}{=} \langle 0, \mathtt{C.SK}, \mu'_0 = \oslash, S_0.\mathtt{SK} = \oslash, \mathtt{CPK} = \oslash\rangle)$

$\mathcal{S}.\mathbf{SUpd}(\mathcal{S}.\mathtt{SK}_t, \mathtt{c})$:
  Let $\mathcal{S}.\mathtt{SK}_t = \langle t, \mathtt{C.SK}, \mu'_t, S_t.\mathtt{SK}, \mathtt{CPK}\rangle$;
  Securely delete $S_t.\mathtt{SK}$;
  $(S_{t+1}.\mathtt{SK}, S_{t+1}.\mathtt{PK}) \leftarrow \Sigma.\mathbf{KeyGen}(1^k)$;
  `if` $(\mathtt{c} = \mathtt{sk\&pk})$ `then`
    $(\mathtt{C.SK}, \mathtt{C.PK}) \leftarrow \Sigma.\mathbf{KeyGen}(1^k)$;
    $\mu_t \leftarrow \langle \mu'_t, \mathtt{C.PK}\rangle$; $\mu'_t \leftarrow \oslash$
  $\mathtt{CPK} \leftarrow \langle S_{t+1}.\mathtt{PK}, \mathtt{C}.\mathbf{Sign}(S_{t+1}.\mathtt{PK})\rangle$;
  $\mu'_{t+1} \leftarrow \mu'_t||S_{t+1}.\mathtt{PK}$
  $\mathcal{S}.\mathtt{SK}_{t+1} = \langle t+1, \mathtt{C.SK}, \mu'_{t+1}, S_{t+1}.\mathtt{SK}, \mathtt{CPK}\rangle$;
  `if` $(\mathtt{c} = \mathtt{sk\_only})$ `then return` $(\mathcal{S}.\mathtt{SK}_{t+1})$;
  `elseif` $(\mathtt{c} = \mathtt{sk\&pk})$ `then return` $(\mathcal{S}.\mathtt{SK}_{t+1}, \mu_t)$;

[8] In principle, C could use a different signature scheme; in some cases, the number of times each instance of C will be used is known—for these cases each C could be an instance of an n-times signature scheme, for the appropriate n.

$\mathcal{S}$.**PUpd**$(\mathcal{S}.\mathtt{PK}_t, \mu_t)$:
Let $\mathcal{S}.\mathtt{PK}_t = \langle \vec{\mathtt{PK}}_i, \mathtt{C.PK} \rangle$; $\mu_t = \langle \vec{\mathtt{PK}}_{[i+1,t]}, \mathtt{C'.PK} \rangle$
`return` ($\mathcal{S}.\mathtt{PK}_{t+1} \stackrel{def}{=} \langle \vec{\mathtt{PK}}_t, \mathtt{C'.PK} \rangle$) % $\vec{\mathtt{PK}}_t = \vec{\mathtt{PK}}_i || \vec{\mathtt{PK}}_{[i+1,t]}$

$\mathcal{S}$.**Sign**$(\mathcal{S}.\mathtt{SK}_t, m)$:
Let $\mathcal{S}.\mathtt{SK}_t = \langle t, \mathtt{C.SK}, \mu'_t, S_t.\mathtt{SK}, \mathtt{CPK} \rangle$;
`return` ($sig = \langle t, \mathtt{CPK}, S_t.$**Sign**$(m) \rangle$)

$\mathcal{S}$.**Ver**$(\mathcal{S}.\mathtt{PK}_t, m, sig)$:
Let $\mathcal{S}.\mathtt{PK}_t = \langle \vec{\mathtt{PK}}_j, \mathtt{C.PK} \rangle$; $sig = \langle i, \mathtt{CPK} = \langle \mathtt{PK}, cs \rangle, s \rangle$
`if` $i \le j$ `then return` ($S_i.$**Ver**(m)) % $S_i.\mathtt{PK}$ *used in the verification is contained in* $\vec{\mathtt{PK}}_j$
`else return` ($\mathtt{C.}$**Ver**$(\mathtt{C.PK}, \mathtt{PK}, cs) \bigwedge \Sigma.$**Ver**$(\mathtt{PK}, m, s)$)

Performance. We use $|PK_\Sigma|$ to denote the size of the public key for the Σ scheme, $|s_\Sigma|$ for the size of the signature under the Σ scheme (for simplicity, we assume that it does not depend on the message signed), l_t for the size of the representation of the time period t, and $|SK_\Sigma|$ is the size of the secret key for Σ. Let t be an arbitrary time period and j be the first time period which has no public key updates between itself and t: $j = \min\{i : P(i) = P(t)\}$. Then our *GKS* scheme above has the following performance characteristics:

The size of public key at time t is $|\mathcal{S}.\mathtt{PK}_t| = j \cdot |PK_\Sigma|$.

All signatures *sig* have length $|sig| = 2|s_\Sigma| + |PK_\Sigma| + l_t$ independent of the time of generation.

The size of the secret key at time t is $|\mathcal{S}.\mathtt{PK}_t| = l_t + 2|SK_\Sigma| + |s_\Sigma| + (t - j + 1) \cdot |PK_\Sigma|$.

The time complexities of all the $\mathcal{S}$ functions are independent of t:

$\mathcal{S}$.**KeyGen** requires only a single Σ.**KeyGen**.

$\mathcal{S}$.**Sign** requires only a single Σ.**Sign**. And $\mathcal{S}$.**Ver** needs at most two Σ.**Ver** computations.

$\mathcal{S}$.**SUpd** requires at most two Σ.**KeyGen** operations, a single Σ.**Sign** and some trivial data (list) manipulations.

$\mathcal{S}$.**PUpd** has O(1) time complexity — independent of Σ complexities, as well as of t.

4.2 Concatenation Scheme Security in the Full Reveal Model

Without essential loss of generality, let the clearing period δ below be $\delta = 1$.

Theorem 1. *Let Σ be an ordinary signature scheme secure against adaptive chosen-message attack with probability of forgery $\le \frac{1}{T2^k}$.*
Then the GKS scheme $\mathcal{S}$ above (Sec. 4.1) is secure: for any PPT forger F',

$$\mathsf{Prob}[Forge_{\mathcal{S}}(F', k, T, \delta) = 1] \le \frac{1}{2^k}$$

Proof sketch: Suppose for some forger F', $\mathsf{Prob}[Forge_{\mathcal{S}}(F', k, T, \delta) = 1] \ge \frac{1}{2^k}$. Then we can construct another forger F who uses adaptive chose-message to attack Σ and succeeds with probability greater than $\frac{1}{T2^k}$.

Forger F guesses uniformly the time period $i : 1 \le i \le T$ for which F' will issue her forgery (that is the time period i in the experiment $Forge_{GKS}$ in Sec. 1). Then F substitutes the public key $\Sigma.\mathtt{PK}$ for which it is trying to obtain the forgery into the $\mathcal{S}.\mathtt{PK}_i$: $S_i.\mathtt{PK} \leftarrow \Sigma.\mathtt{PK}$. All the other keys are generated by F according to the $\mathcal{S}$ algorithm.

The probability that F chooses this same i is $1/T$. If the guess was wrong, F aborts. If the guess was correct then with the probability $\ge \frac{1}{2^k}$, F' outputs a signature and a message valid for $S_i.\mathtt{PK} = \Sigma.\mathtt{PK}$, which were never queried, nor the corresponding key was queried.

Thus, F is successful with the probability that it makes the right guess $(1/T)$ and F' succeeds $(\ge \frac{1}{2^k})$. Therefore the probability of F succeeding is $\ge \frac{1}{T2^k}$. □

4.3 Partial Reveal Schemes

A Simple Trade-off. The schemes of [18] all use the partial reveal model. Still, they have both public keys and signatures linear in the maximum number of public key updates T_P.

In our full reveal scheme, the signature size is reduced to constant while the public key size is linear in the current time period.[9]

Furthermore, the size of the signatures in the schemes of [18] gives some indication to the attacker of what T_P might be equal to. Thus the attacker may force the signer to reveal all or most of the secrets in one attack. In contrast our concatenation scheme has no T_P and thus can be continued indefinitely, and the signer has no hidden secrets for the attacker to extort — all the secrets to be used in the future are generated as needed.

We note that the schemes of [18] can be directly optimized to reduce the public key size to constant, while keeping the signature size linear in T_P; the verification would also be reduced to constant — one ordinary signature verification. Indeed, intuitively each monotone signature is verified with respect to all the published public key components. But for security, it is sufficient to verify the signatures only against the last public key component published. Thus the previously published public keys can be deleted. Of course, for this optimized scheme it is possible to generate signatures that could be repudiated at any time later on. This scheme also allows the signer to violate monotonicity: generate signatures which are invalid at the time of generation, but valid at the later times.

In fact, combining similar optimization with our concatenation scheme, it is possible to achieve a linear trade-off between the signature and public key sizes.

[9] To be fair, we must note that in our scheme the number of updates may be larger than in the monotone signatures: since we update the secret key every time there is a threat of exposure. On the other hand, this also allows us to deal with full-reveal inconspicuous exposures, which the monotone signatures are unable to protect against. If inconspicuous exposures are not a threat, then the `sk_only` updates can be implemented using a forward-secure signature scheme Σ, reducing the size of the public key to (nearly) match the monotone signatures.

However, such trade-off is only of theoretical value, since it is very unlikely that the savings of the public key length might justify the proportional increase of the signature length.

Interval Crossing Scheme (*IX*)

Intuition. This section presents a *GKS* scheme for the partial reveal model exponentially improving the asymptotic performance of the above schemes, as well as the schemes of [18]. We call it *interval crossing scheme* or *IX* for short.

IX is based on the forward-secure scheme of [13]. It also uses two security parameters: k and l. Intuitively, k is the length of the modulus used for GQ-like signatures [9]; l is the length of the outputs of the random oracle used for the scheme.

Let T be an upper-bound on the number of all the key updates. The scheme of [13] divides the interval $[2^l, 2^{l+1}]$ into T subintervals $I_t = [2^l + (t-1)2^l/T,\ 2^l + t2^l/T)$ — one per each time period t. A prime exponent $e_t \in I_t$ is used in the GQ-like the signatures for the time period t. *IX* further subdivides each I_t into C intervals, for the parameter C of the given *IX* scheme. It then offers an option to specify the sub-interval of e_t with greater precision: The *IX* public key may include index $1 \leq i_t \leq C$ for each time period t; if the exponent e_t used in a signature for time t is not from the i_t-th subinterval of I_t, then the signature is invalidated. For convenience we identify the index i_t with the i_t-th subinterval of I_t.

Intuitively, C corresponds to the upper-bound on the number of ways the signer may try to cheat the attacker. While C is a parameter of the scheme, and is thus assumed to be known to the attacker, the actual number of the spare versions of e_t prepared for each interval by the signer is unknown. Indeed, as will become apparent from below, it may be wise to select both T and C to be fairly large — the cost to the signer is only logarithmic in C (recording the index i_t in the public key) and independent of T. In fact these two parameters may be set for some fixed large values, same for all users (e.g., $T = 10^{10}$, $C = 2^{10}$).

Thus, if no extortion attacks took place in period t, then no index i_t for that period needs to be published in the *IX* public key. However, if the signer is attacked during the time t then he can reveal some secrets for one or more $e'_t \in I_t$, leaving at least one $e_t \in I_t$ not revealed. Upon release, the signer can update the public key, publishing i_t; all signatures for time t using $e'_t \notin i_t$ are then disqualified. Clearly, if the indistinguishability of the "fake" and "real" secrets in this case can be obtained only if the attacker does not have any legitimate signatures for the same period t. This can be achieved, for example, by the signer performing an update (thus incrementing t) as soon as he comes under the attack.

This mechanism is essentially similar to "black-listing" the extorted keys (more accurately, "white-listing" a non-extorted key) when needed. Applied directly, this method would not offer significant improvement over the concatenations scheme. However we achieve an exponential improvement by using a compact representation of the "white-list".

Compact set representation. Consider the following problem stated here informally: We need to generate a set S of t_{max} elements so that for any $t \le t_{max}$, the subset $S_t \subseteq S$ of t elements can be represented compactly and in such a way that no information about $S - S_t$ is revealed. In particular, no information about the size of S is leaked, nor is it possible to deduce whether a given $x \notin S_t$ is actually in S.

We propose the following, slightly simplified approach: Let most of the elements of the sets S_t be leaves of a small (logarithmic in t) number of trees (e.g., at most one of each height). Each tree can be represented by its height and the value at its root. All the tree leaves are computed from the root as in the pseudo-random function (PRF) tree of [7]. A small —bounded by constant— number of elements of S_t are not leaves in such trees, but are listed individually.

We use this approach as follows: as the i_t values are published in the IX public key, they are initially listed individually. As more of them are collected, they are aggregated into the trees. At any moment of time, the set of the revealed i_t's gives no indication of whether it has reached t_{max}, the maximum number of periods for which the signer had prepared the secret key. Moreover, the latest revealed "fake" secrets cannot be distinguished from the "real" ones.

IX-Scheme: Formal Description. For the sake of simplicity in describing the algorithms, we assume that $C = 2$. Thus, each i_t can be specified with a single bit (in addition to specifying t). Let $H : \{0,1\}^* \to \{0,1\}^l$ be a random function, $\Delta \stackrel{def}{=} 2^l/(TC)$, and t_{max} be the maximum number of the time periods for the give scheme instance. Let P be a length-doubling PRG, $P : \{0,1\}^L \to \{0,1\}^{2L}$, wlog assume $L = l$. We use P to generate PRF trees. Consider a balanced PRF tree with T leaves, and let R_t be the smallest set of the tree nodes such that the set of the maximal subtrees rooted in R_t contains exactly the leftmost t leaves of the tree, and contains no trees with exactly two leaves and at least one tree of size one. Let the j-th leftmost leaf determine i_j. Say $(e, j) \in R_t$ whenever $e \in i_j$ as determined by R_t.

KeyGen (k, l, T)

Generate random $(\lceil \frac{k}{2} \rceil - 1)$-bit primes q_1, q_2 s.t. $p_i = 2q_i + 1$ are both primes.

$n \leftarrow p_1 p_2$

For a PRF tree of depth $\lceil \lg T \rceil$, generate $R_{t_{max}}$.

For each $t : 1 \le t \le t_{max}$, generate primes e_t, e'_t such that $e_t \in i_t$ and $e'_t \in I_t - i_t$.

(let $f_i \stackrel{def}{=} e_i e'_i \dots e_{t_{max}} e'_{t_{max}}$, $F'_i \stackrel{def}{=} f_{i+1} \cdot e'_i$, $F_i \stackrel{def}{=} f_{i+1} \cdot e_i$, and $\vec{e}_{[i,j]} \stackrel{def}{=} \langle e_i, e'_i, \dots, e_j, e'_j \rangle$)

$b_1 \stackrel{R}{\leftarrow} Z_n^*$

$v \leftarrow 1/b_1^{f_1} \bmod n$

$s_1 \leftarrow b_1^{F'_1} \bmod n$

$s'_1 \leftarrow b_1^{F_1} \bmod n$

$b_2 \leftarrow b_1^{e_1 e'_1} \bmod n$

$\mathtt{SK}_1 \leftarrow \langle R_{t_{max}}, 1, t_{max}, n, s_1, e_1, b_2, \vec{e}_{[2,t_{max}]} \rangle$ % *real secret*

$\mathtt{SK}'_1 \leftarrow \langle \neg R_1, 1, 1, n, s'_1, e'_1, v, \vec{e}_{[2,1]} = \oslash \rangle$ % *fake secret, to be revealed in key exposure*

$\mathtt{PK}_1 \leftarrow (n, v, \oslash)$

return $(\mathtt{SK}_1, \mathtt{SK}'_1, \mathtt{PK}_1)$ % *while formally* $\mathtt{SK}'$*, should be part of* SK *we keep them separate for clarity; the signer disposes of* $\mathtt{SK}'$ *before issuing the first signature of the period*

SUpd (SK_j)

Let $\mathtt{SK}_j = \langle R_{t_{max}}, j, t_{max}, n, s_j, e_j, b_{j+1}, \vec{e}_{[j+1,t_{max}]} \rangle$

if $j = t_{max}$ then **return** $\oslash$

$s_{j+1} \leftarrow b_{j+1}^{F'_{j+1}} \bmod n$;

$s'_{j+1} \leftarrow b_{j+1}^{F_{j+1}} \bmod n$;

$b_{j+2} \leftarrow b_{j+1}^{e_{j+1}e'_{j+1}} \bmod n$

return $\mathtt{SK}_{j+1} = \langle R_{t_{max}}, j+1, t_{max}, n, s_{j+1}, e_{j+1}, b_{j+2}, \vec{e}_{[j+2,t_{max}]} \rangle$ and fake secret key

$\mathtt{SK}'_{j+1} = \langle R_j \cup e'_{j+1}, j+1, n, s'_{j+1}, e'_{j+1}, v, \oslash \rangle$;

$\mu \leftarrow R_j$; % *it is sufficient to include the part of* R_j *only for the periods of extortion*

PUpd $(\mathtt{PK}_j, \mu_j)$

Let $\mathtt{PK}_j = \langle n, v, \ldots \rangle$.

return $\mathtt{PK}_{j+1} = \langle n, v, \mu \rangle$

Sign (SK_j, M)

let $\mathtt{SK}_j = \langle R_{t_{max}}, j, t_{max}, n, s_j, e_j, b_{j+1}, \vec{e}_{[j+1,T]} \rangle$

$r \overset{R}{\leftarrow} Z_n^*$

$y \leftarrow r^{e_j} \bmod \mathrm{n}$

$\sigma \leftarrow H(j, e_j, y, M)$

$z \leftarrow r s_j^{\sigma} \bmod \mathrm{n}$

return (z, σ, j, e_j)

Ver $(\mathtt{PK}_t, M, (z, \sigma, j, e))$

Let $\mathtt{PK}_t = \langle n, v, R_{t'} \rangle$.

if e is even **or** $e \notin I_j$ **or** $z \equiv 0 \bmod n$ **then return** 0

if $j \leq t'$ **and** $(e, j) \notin R_{t'}$ **then return** 0

$y' \leftarrow z^e v^{\sigma}$

if $\sigma = H(j, e, y', M)$ **return** 1 **else return** 0

Reveal

When face dangers, the signer just needs to update his secret key and begins new time period, j. For time period j, the signer reveals the fake secret key $\mathtt{SK}'_j$.

The proof of security of the above scheme is similar to the proof in [13]. Intuitively, if the forger breaks in some time period, the signer can reveal fake secret key for that time period, this fake secret is indistinguishable from the real secret key under the current public key. Furthermore, this fake secret key doesn't help the forger to guess the real secret key by the variant of the strong RSA assumption in [4]. The probability that the forger succeeds in generating valid signature for other time periods is negligible. In the above algorithm, we always use the fixed fake secret. Actually, the signer can reveal several fake secret

keys. The attacker have no idea how many secret keys in each interval because the T and s is unknown to the attacker, and the number of secret candidates is probabilistic.

From the construction, we can see that the size of the signature is constant and the size of the public key grows logarithm with time. This is in contrast to the full reveal scheme where each time period added the security parameter number of bits.

5 Lower Bounds: Full Reveal Model

In this section we consider the *GKS* schemes in the full reveal model, and prove the lower bound for the public key length matching that of our scheme in sec. 4.1.

First, a simple probability observation:

Claim. For any events A, B, C, $\mathsf{Prob}[A \bigwedge B|C] = \mathsf{Prob}[A|C] \cdot \mathsf{Prob}[B|A \bigwedge C]$.

Indeed, starting with the right-hand side and using the conditional probability definition we get $\mathsf{Prob}[A|C] \cdot \mathsf{Prob}[B|A \bigwedge C] = \frac{\mathsf{Prob}[A \bigwedge C]}{\mathsf{Prob}[C]} \cdot \frac{\mathsf{Prob}[B \bigwedge A \bigwedge C]}{\mathsf{Prob}[A \bigwedge C]} = \frac{\mathsf{Prob}[A \bigwedge B \bigwedge C]}{\mathsf{Prob}[C]} = \mathsf{Prob}[A \bigwedge B|C]$. □

Now, let $\mathcal{S}$ be an *GKS*-secure scheme and F be a forger. Fix some time period t, immediately after a public key update.[10] For $i \leq t$, let f_i denote an event that an attacker generates a signature valid for $\mathcal{S}.\mathtt{PK}_t$, and some message and time period (m, i) pair (without querying the $Osig(m, i)$ oracle). Let b_i be the event of the attacker break-in at period i (in other words, a query to $Orvl_i$ oracle); assume no other key exposures except those mentioned explicitly. Recall that k is the security parameter and $\mathsf{Prob}[f_i|\neg b_i] \leq 1/2^k$ (see definition 1).

Lemma 1. $\mathsf{Prob}[(f_1 \wedge f_2 \wedge \ldots f_t)|b_0] \leq \frac{1}{2^{kt}}$, *where k is the security parameter.*

Proof: $\mathsf{Prob}[(f_1 \wedge f_2 \wedge \ldots f_t)|b_0]$
$= \mathsf{Prob}[f_1|b_0] \times \mathsf{Prob}[(f_2 \wedge f_3 \ldots f_t)|(f_1 \wedge b_0)]$ (by the claim above)
$\leq \mathsf{Prob}[f_1|b_0] \times \mathsf{Prob}[(f_2 \wedge f_3 \ldots f_t)|(b_1 \wedge b_0)]$
$= \mathsf{Prob}[f_1|b_0] \times \mathsf{Prob}[f_2|(b_1 \wedge b_0)] \times \mathsf{Prob}[(f_3 \wedge f_4 \ldots f_t)|(f_2 \wedge b_1 \wedge b_0)]$
$\vdots$
$\leq \mathsf{Prob}[f_1|b_0] \times \mathsf{Prob}[f_2|(b_1 \wedge b_0)] \times \mathsf{Prob}[(f_3|b_2 \wedge b_1 \wedge b_0] \ldots \mathsf{Prob}[f_t|(b_{t-1} \wedge b_{t-2} \ldots b_0)]$
$\leq \frac{1}{2^k} \times \frac{1}{2^k} \ldots \frac{1}{2^k} = \frac{1}{2^{kt}}$ by the definition of *GKS*. □

For the full reveal *GKS* signatures, we assume that the attacker receives all the secrets of the system at the time period of the break-in. Thus immediately after b_i, the real signer and the attacker F have exactly the same information. The difficulty of $f_{j>i}$ relies on the fact that the evolutions of the signer and F

[10] The next lemma, and thus the subsequent theorem, rely on the assumption that the clearing period is $\delta = 1$. Adjusting them to arbitrary δ would require that we talk below about the public key $\mathtt{PK}_{t'}$ such that t' is δ public key updates after t: $P(t') \geq P(t) + \delta$. Then the lower bound would be shown for the public key $\mathtt{PK}_{t'}$ instead of $\mathtt{PK}_t$. The rest of the section would remain intact.

are likely to diverge. If somehow F, emulating the evolution of the real signer, arrives at the same public key $\mathcal{S}.\mathtt{PK}_j$ as the real signer, then F can generate signatures for all messages and all the time periods $i' : i \leq i' \leq j$ and valid for $\mathcal{S}.\mathtt{PK}_j$.

Thus, probability of F emulating real signer evolution and arriving at the same public key $\mathcal{S}.\mathtt{PK}_t$ as the real signer is no greater than $\mathsf{Prob}[(f_1 \wedge f_2 \wedge \ldots f_t)|b_0]$. Which means that this probability of evolving into $\mathcal{S}.\mathtt{PK}_t$ is $\leq \frac{1}{2^{kt}}$. But since both signer and F use the same probability distributions, this implies the following theorem:

Theorem 2. *Let $\mathcal{S}$ be a GKS signature scheme secure (with security parameter k) in the full reveal model and let t immediately follow a public key update. Then $|\mathcal{S}.\mathtt{PK}_t| \geq kt$*

The theorem implies optimality of our scheme in Sec. 4.1 at least with respect to the length of the public keys.

Acknowledgments. The authors are grateful to Shai Halevi for helpful discussions.

References

1. *Third Conference on Security in Communication Networks (SCN'02)*, Lecture Notes in Computer Science. Springer-Verlag, September 12–13 2002.
2. Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129, Kyoto, Japan, 3–7 December 2000. Springer-Verlag. Full version available from the Cryptology ePrint Archive, record 2000/002, `http://eprint.iacr.org/`.
3. Ross Anderson. Invited lecture. Fourth Annual Conference on Computer and Communications Security, ACM (see http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf), 1997.
4. Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology—EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, 11–15 May 1997.
5. Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer-Verlag, 15–19 August 1999. Revised version is available from `http://www.cs.ucsd.edu/~mihir/`.
6. Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. Unpublished Manuscript.
7. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.
8. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

9. Louis Claude Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 1990, 21–25 August 1988.
10. Johan Håstad, Jakob Jonsson, Ari Juels, and Moti Yung. Funkspiel schemes: an alternative to conventional tamper resistance. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 125–133. ACM Press, 2000.
11. Gene Itkis. Cryptographic tamper evidence. Submitted. Avaliable from `http://www.cs.bu.edu/ itkis/papers/`, 2002.
12. Gene Itkis. Intrusion-resilient signatures: Generic constructions, or defeating strong adversary with minimal assumptions. In *Third Conference on Security in Communication Networks (SCN'02)* [1].
13. Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354. Springer-Verlag, 19–23 August 2001.
14. Gene Itkis and Leonid Reyzin. Intrusion-resilient signatures, or towards obsoletion of certificate revocation. In Moti Yung, editor, *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Computer Science. Springer-Verlag, 18–22 August 2002. Available from `http://eprint.iacr.org/2002/054/`.
15. Anton Kozlov and Leonid Reyzin. Forward-secure signatures with fast key update. In *Third Conference on Security in Communication Networks (SCN'02)* [1].
16. Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In *Seventh ACM Conference on Computer and Communication Security*. ACM, November 1–4 2000.
17. Tal Malkin, Daniele Micciancio, and Sara Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Lars Knudsen, editor, *Advances in Cryptology—EUROCRYPT 2002*, Lecture Notes in Computer Science. Springer-Verlag, 28 April–2 May 2002.
18. David Naccache, David Pointcheval, and Christophe Tymen. Monotone signatures. In P. Syverson, editor, *Financial Cryptography*, volume 2339 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 2001.

A Ring Signature Scheme Based on the Nyberg-Rueppel Signature Scheme

Chong-zhi Gao[1], Zheng-an Yao[2], and Lei Li[1]

[1] Institute of Software
Sun Yat-Sen University
Guangzhou 510275, China
[2] College of Mathematics and Computational Science
Sun Yat-Sen University
GuangZhou 510275, China

Abstract. Ring signature allows to specify a set of possible signers without revealing which member actually produced the signature. This concept was first formalized in 2001 by Rivest, Shamir, and Tauman[3]. Authors of [3] also proposed two versions of ring signature scheme. However, to achieve the goal of anonymity, each user should do much computation in the initial procedure: they should do much work to generate their private and public keys, e.g. in the RSA version, each user should find n_i such that it is the product of two distinct large prime and compute his private/public keys. Moreover, one should extend the one-way trapdoor functions to a common domain since these functions are computed in different domains. This paper's main contribution is to present a version of ring signature scheme which uses a common modulus. Thus, Our proposed scheme is much more efficient in the setup procedure. Further more, the size of public and private keys are reduced.

1 Introduction

Ring signature scheme was first formalized in 2001 by Rivest, Shamir, and Tauman[3]. A ring signature makes it possible to specify a set of possible signers without revealing which member actually produced the signature.

To achieve the goal of anonymity, each user should generate his public key and private key independently, i.e. there does not exist a trust center that do the initial work such as generating secret keys and transmitting these keys to users. For example, in the RSA version of the ring signature, each user should select n_i such that it is the product of two distinct large prime. Then he compute his public and private key. Obviously, this initial work involves much computation. Further more, it require the signer to do additional preparation when he generate the signature. Similarly, this happens in the Rabin version.

Nyberg and Rueppel [1,2] proposed a signature scheme based on the discrete logarithm problem. The Nyberg-Rueppel scheme allows signatures with message recovery. Our paper's main contribution is to present a ring signature scheme with a common modulus, i.e. all users compute in the domain $\mathbf{Z}_p$ where

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 169–175, 2003.

p is a large prime. Applying the Nyberg-Rueppel scheme to the ring signature, without weakening the security, different users need not do the computation in different domains. Thus, our proposed scheme is much more efficient in the setup procedure. Further more, the size of keys (public and private) are reduced.

1.1 Related Works

Chaum and van Heyst put forth the concept of a group signature scheme In 1991 ([4]). It allows a group member to sign messages anonymously on behalf of the group. However, the identity of the signature's originator can be revealed by the group manager if necessary. A group signature scheme could be used by an employee of a large company to sign documents on behalf of the company and it has further applications. Informal notions of ring signatures which was in the context of general group signatures can be finded in [4, 5]. However, the concept of ring signatures was first formalized in [3]. A threshold ring signature scheme was proposed in [6] by Emmanuel Bresson, Jacques Stern, and Michael Szydlo.

The rest of this paper is organized as follows. In section 2, we explain ring signatures and review some previous works. In section 3, we introduce our ring signature scheme and discuss its security and efficiency. In section 4, we conclude the paper.

2 Ring Signature

2.1 Ring Signature

The concept of ring signature was formalized by Rivest, Shamir, and Tauman in [3].

It is assumed that each possible signer is associated with a public key P_k that defined his signature scheme and specifies his verification key. The corresponding secret key (which is used to generate regular signature) is denoted by S_k. These possible signers are called a *ring*. It is also necessary that it is a trapdoor one-way function to generate and verify signatures.

A ring signature consists two procedures:

– ring-sign(m, P_1, P_2,..., P_r, s, S_s) which produces a ring signature σ for the message m, given the public keys P_1, P_2,..., P_r of the r ring members, together with the secret key S_s of s-th member (who is the actual signer).

– ring-verify(m, σ) which accepts a message m and a signature σ (which includes the public keys of all the possible signers), and outputs either *true* or *false*.

A ring signature scheme must satisfy the usual soundness and completeness condition. In addition the signatures should be *signer-ambiguous* : the verifier should be unable to determine the identity of the actual signer in a ring of size r with probability greater than $1/r$.

The scheme proposed in [3] has the property of unconditional anonymity in the sense that even an infinitely powerful adversary can not reveal the actual signer's identity.

2.2 Previous Work

To compare with our proposed method, we review the method in [3].

Combining Functions:

The concept of combining functions is very important in the ring signature scheme.

A Combing function $C_{k,v}(y_1, ..., y_r)$ takes as input a key k, an initialization value v, and arbitrary values $y_1, y_2, ..., y_r$ in $\{0,1\}^b$ such that given any fixed values for k and v, it has the following properties:

1. **Permutation on each input:** For each s, $1 \leq s \leq r$, and for any fixed values of all the other inputs y_i, $i \neq s$, the function $C_{k,v}$ is a one-to-one mapping from y_s to the output z.

2. **Efficiently solvable for any single input:** For each s, $1 \leq s \leq r$, given a b-bit value z and values for all inputs y_i except y_s, it is possible to efficiently find a b-bit value for y_s such that $C_{k,v}(y_1, ..., y_r) = z$.

3. **Infeasible to solve verification equation for all inputs without trap-doors:** Given k, v and z, it is infeasible for an adversary to solve the equation

$$C_{k,v}(g_1(x_1), ..., g_r(x_r)) = z$$

for $x_1, x_2, ..., x_r$, (given access to each g_i, and to E_k) if the adversary can't invert any of the trap-door function $g_1, g_2, ..., g_r$.

[3] proposed a combining function:

$$C_{k,v}(y_1, ..., y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(... \oplus E_k(y_1 \oplus v...))))$$

where E_k is a symmetric encryption function which takes k as secret key.

It satisfies the three properties and the y_s is computed as follows:

$$y_s = E_k^{-1}(y_{s+1} \oplus ...E_k^{-1}(y_r \oplus E_k^{-1}(z))) \oplus E_k(y_{s-1} \oplus ...E_k(y_1 \oplus v)...)$$

Trap-Door Permutations and Extending Them to a Common Domain:

Each member $A_i (1 \leq i \leq r)$ generates his RSA key $k_i = (n_i, p_i, q_i, e_i, d_i)$. Where $P_i = (n_i, e_i)$ is his public key and $S_i = (p_i, q_i, d_i)$ is his private key.

The one-way permutation f_i of Z_{n_i} is defined as:

$$f_i(x) = x^{e_i} \pmod{n_i}$$

The inverse permutation f_i^{-1} is computed as:

$$f_i^{-1}(y) = y^{d_i} = x \pmod{n_i}.$$

However, the trap-door RSA permutations of the various ring members have domains of different sizes. The authors in [3] extend the permutations to have as their common domain the same set $\{0,1\}^b$, where 2^b is some power of two which is larger than all the modular n_i's.

The extended trap-door permutation g_i over $\{0,1\}^b$ is defined as:

For any b-bit input m define nonnegative integers t_i and r_i so that $m = t_i n_i + r_i$ and $0 \leq r_i < n_i$. Then

$$g_i(m) = \begin{cases} t_i n_i + f_i(r_i) & \text{if } (t_i+1)n_i \leq 2^b \\ m & else \end{cases}$$

For more details, please consult [3].

The Ring Signature of RSA Version in [3]:

Given the message m to be signed, the signer's secret key S_s, and the sequence of public keys $P_1, P_2, ..., P_r$ of all the ring members, the signer computes a ring signature as follows:

1. The signer computes $h(m)$ as the symmetric key k:

$$k = h(m)$$

2. The signer picks an initialization value v uniformly at random from $\{0,1\}^b$.

3. The signer picks random x_i for all the other ring members $1 \leq i \leq r$, $i \neq s$ uniformly and independently from $\{0,1\}^b$, and computes

$$y_i = g_i(x_i)$$

4. The signer solves the following ring equation for y_s:

$$C_{k,v}(y_1, y_2, ..., y_r) = v.$$

By assumption, given arbitrary values for the other inputs, there is a unique value for y_s satisfying the equation, which can be computed efficiently.

5. The signer uses his knowledge of his trapdoor in order to invert g_s on y_s to obtain x_s:

$$x_s = g_s^{-1}(y_s).$$

6. The signature on the message m is defined to be the $(2r+1)$-tuple:

$$(P_1, P_2, ..., P_r; v; x_1, x_2, ..., x_r).$$

To verify a signature

$$(P_1, P_2, ..., P_r; v; x_1, x_2, ..., x_r)$$

on the message m, the verifier accepts the signature if and only if $C_{H(m),v}(f_1(x_1), f_2(x_2), ..., f_r(x_r))$ is equal to v.

3 Our Proposed Scheme – Nyberg-Rueppel Version

To achieve the goal of using common modulus when computing, we observe that the RSA signature scheme is not suitable for this goal. We also point out that general signature schemes is not suitable since it should be a trapdoor one-way function to generate and verify signatures. For example, the ElGamal signature scheme is not suitable. We should employ the signature schemes that anyone can generate "pseudo" signatures which can be successfully verified using the verification procedure. But only a person who has his secret key can generate a "real" signature of a message by his own choice.

Considering the Nyberg-Rueppel signature scheme, we find that it meet our requirement and thus can be applied to the ring signature scheme.

We build the trapdoor one-way functions based on the Nyberg-Rueppel signature scheme. Let's describe the ring signature scheme of Nyberg-Rueppel version in detail.

Combining Functions:
We use the same combining function proposed in [3], which was described in part 2.2 in our paper.

Trapdoor Functions using common modulus:
Let p be a prime such that the discrete log problem in $\mathbf{Z}_p$ is intractable. Let $\alpha \in GF(p)$ be an element of order q where q is equal to $p-1$ or is a large integer factor of $p-1$. (p, α) are the common parameters of all the ring members.

Each user A_i chooses his private key $S_i \in \mathbf{Z}_q$ and publishes his public key as $P_i = \alpha^{-S_i} \bmod p$.

The one-way function f_i takes a 2-tuple $(e, y) \in \mathbf{Z}_p \times \mathbf{Z}_q$ as input and output a single number in $\mathbf{Z}_p$. It is computed as follows:

$$f_i(e, y) = \alpha^y {P_i}^e e \bmod p$$

If one know the value S_i, one can find an inverse of x $(= f(e, y))$ by following steps:

First, randomly chooses $r \in \mathbf{Z}_p$, then computes:

$$e = x\alpha^{-r} \bmod p$$

$$y = r + S_i e \bmod q$$

and output (e, y).

This one-way function is derived from the Nyberg-Rueppel signature scheme. Note that there are not only one inverse of an x. Everyone can compute f_i using the public key P_i, but given $x \in GF(p), x \neq 0$, it is hard for anybody not in possession of S_i to compute an inverse of x, since it requires the solution of the equation:

$$x = \alpha^y {P_i}^e e \bmod p$$

for y and e. The security of Nyberg-Rueppel signature scheme was analyzed in [1] and [2].

The Ring Signature of Nyberg-Rueppel version:
If a signer who has the secret key S_s want to sign a message m, and the public keys of all the ring members are $P_1, P_2, ..., P_r$. The singer computes a ring signature as follows.

1. The signer computes $h(m)$ as the symmetric key k:

$$k = h(m)$$

2. The signer picks an initialization value v uniformly at random from $\{0,1\}^b$.
3. The signer picks random (e_i, y_i) uniformly and independently from $\mathbf{Z}_p \times \mathbf{Z}_q$ $(1 \leq i \leq r, i \neq s)$ and computes:

$$x_i = f_i(e_i, y_i)$$

4. The signer solves the following ring equation for x_s:

$$C_{k,v}(x_1, x_2, ..., x_r) = v$$

5. The signer uses his secret key S_i to invert f_s on x_s to obtain (e_s, y_s).
6. The signature on the message m is defined to be the $(2r+1)$-tuple:

$$(P_1, P_2, ..., P_r; v; (e_1, y_1), ..., (e_r, y_r))$$

To verify a signature

$$(P_1, P_2, ..., P_r; v; (e_1, y_1), ..., (e_r, y_r))$$

on the message m, the verifier accept the signature if and only if

$$C_{H(m),v}(f_1(e_1, y_1), f_2(e_2, y_2), ..., f_r(e_r, y_r))$$

is equal to v.

Security and Efficiency:

We use the combining function proposed in [3] and refer the reader to [3] for the analysis of the combining function's security. Now, Let's consider the one-way functions derived from Nyberg-Rueppel signature scheme. Everyone can compute f_i provided with the public key P_i. However, given the value of $f_i(e, y)$, only the one who has the secret key S_i can obtain (e_0, y_0) such that $f_i(e, y) = f_i(e_0, y_0)$. Hence, our proposed scheme satisfies the properties of completeness and soundness.

Our proposed scheme also has the property of unconditional anonymity, i.e. even an infinitely powerful adversary can not reveal the actual signer's identity, because given m and v, the function

$$C_{H(m),v}(f_1(e_1, y_1), f_2(e_2, y_2), ..., f_r(e_r, y_r))$$

has $p^{r-1}q^r$ solutions for $((e_1, y_1), (e_2, y_2), ..., (e_r, y_r))$. Every one of these solutions might be chosen with equal probability by an signer.

Let us take a look at the size of keys. The public key of each user is in $\mathbf{Z}_p$ and the private key is in $\mathbf{Z}_p$, whereas in the RSA version, the public key of user A_i is (n_i, e_i) in $\{0,1\}^b \times \{0,1\}^b$ and the private key is (p_i, q_i, d_i) in $\{0,1\}^{b/2} \times \{0,1\}^{b/2} \times \{0,1\}^b$ where $b/2$ is the approximate length of each prime in binary representation. Thus, the size of keys are greatly reduced.

Since the users need not to search large primes to compose their public keys and private keys, there is much less computation in the initial work. But our proposed scheme also has it's drawback: signing requires one modular exponentiation and two modular exponentiation for each non-signer; verification requires two modular exponentiation for each ring member. Thus, this scheme is more suitable in the case of small rings.

4 Conclusions

We have proposed a ring signature scheme using a common modulus. We showed that our method satisfies the property of unconditional anonymity and require much less computation in the setup procedure. Further more, the size of public and private keys are reduced.

References

[1] K. Nyberg and R. Rueppel, A new signature scheme based on the DSA giving message recovery, Proc. 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, Nov. 3–5., (1993), 4 pages.

[2] K. Nyberg and R. Rueppel, Message recovery for signature schemes based on the discrete logarithm problem, in Advances in Cryptology-EuroCrypt'94, LNCS 950, pp.182–193, Springer-Verlag, 1995.

[3] R. Rivest, A. Shamir, and Y. Tauman, How to Leak a Secret, in Proc. Asiacrypt '01, pp. 552–565, Springer-Verlag, 2001.

[4] David Chaum and Eugène Van Heyst, Group signatures, in Advances in Cryptology Eurocrypt 91, pp. 257-C265, Springer Verlag, LNCS 547, 1991.

[5] Jan Camenisch, Efficient and generalized group signatures. In Walter Fumy, editor, Advances in Cryptology C Eurocrypt 97, pp. 465-C479, Springer Verlag, LNCS 1233, 1997.

[6] Emmanuel Bresson, Jacques Stern, Michael Szydlo, Threshold Ring Signatures for Ad-hoc Groups , in Advances in Cryptology – CRYPTO'2002, pp. 465-C480, Springer Verlag, LNCS 2442, 2002.

Modelling and Evaluating Trust Relationships in Mobile Agents Based Systems

Ching Lin and Vijay Varadharajan

Information and Networked System Security Research
Department of Computing
Division of Information and Communication Sciences
Macquarie University
Sydney, NSW 2019, Australia
{linc, vijay}@ics.mq.edu.au

Abstract. This paper considers a trust framework for mobile agent based systems. It introduces the concept of trust reflection and presents a novel method using views of the trustor to derive the trust symmetry. This new approach addresses the trust initialization issues and allows for different levels of initial trust to be established based on trustor's initial assessment and then enables different trust dynamics to evolve in the course of future interactions in situations where full trust appraisal is not possible at the beginning of interactions. This is an important aspect for security in distributed systems, especially in mobile agent based systems due to the fact that the agent owner principals may not have all the information to appraise full trust in other entities(e.g. foreign hosts to be visited by the mobile agent) when composing an itinerary prior to the deployment of mobile agent. Our framework proposes a new formalism to capture and reason about trust reflection and presents an algorithm for updating trust during system operation. This is then applied to a simulated mobile agent system and an analysis of the results is presented.

Keywords: Security, Trust, Mobile Agent System

1 Introduction

Over the recent years, mobile agent technologies have been receiving a great deal of interest, as they have a lot to offer towards achieving the vision of usable distributed systems in a heterogeneous network environment. The ability to move computations across the nodes of a wide area network helps to achieve the deployment of services and applications in a more flexible, dynamic and customizable way than the traditional client-server paradigm. They provide several advantages over remote procedural call and message passing such as reduced network usage, increased asynchrony between clients and servers, increased concurrency and addition of client-specified functionality into servers. However, there lie some fundamental issues that need to be addressed in the area of security. The key to achieving this is the systematic understanding and developing a comprehensive

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 176–190, 2003.

security model for mobile agent systems [1]. Presently, the mobile agent security research focuses primarily on the design of cryptography based security mechanisms for the protection of mobile agents and hosts. Typical solutions include the following: a) for host protection: state appraisal [15], signed code [16] PCC [19]; b) for agent protection: execution tracing [23], computing with encrypted functions [21], hardware based protection [24]. c) combined host and agent security solutions include [18,1]. A more comprehensive summary on mobile agent security can be found in [14,17,18,20].

In this paper, we identify trust as a fundamental component of the mobile agent security architecture and propose a trust framework for mobile agent system. Though trust is often recognized as the foundation of secure systems, it is seldom represented and reasoned about explicitly when it comes to the practical design of secure systems and applications. Our aim in this paper is to represent trust explicitly and reason about it in the context of mobile agent based system operation. In particular, we aim to be able to reason about the trust properties such as the execution trust and mobile code trust, which will enable us to make more informed decisions to achieve our security goals in a mobile agent system.

The paper is organized as follows. Section 2 discusses some security related trust problems in a mobile agent system. It poses six trust questions that should be answered at different stages of the life cycle of a typical mobile agent operation and introduces the trust reflection problem. Section 3 proposes a new formalism to capture and reason about trust reflection and presents an algorithm for updating trust during system operation. Section 3.4 applies the new formalism and the algorithm to the mobile agent security problems introduced in Section 2. Finally, Section 5 concludes with some ideas for further work.

2 Security Related Trust Problems in Mobile Agent System

Conceptually, trust on an entity can be thought of as a combination of honesty, competence, reliability and availability of the entity with respect to interaction and cooperation. In this paper, we refine our approach to modelling trust in mobile agent systems by considering a new form of trust which we call *reflective trust* which we will introduce and discuss in detail in the following sections. Broadly speaking, in the context of mobile agents, we believe two types of trust that need to be addressed to cater for host and mobile code security issues. From the mobile agent owner point of view, we have trust on execution: this is the trust that the hosts will faithfully run mobile code and then migrate it to next destination (see figure 2); this trust is related to the underlying mechanisms for mobile code security, which provide preventive measures using TTPs to verify and certify the hosts' capacity for running the mobile code [4]; this trust is also related to the detection mechanisms such as signed security tags and chained hashing algorithms for mobile agent data integrity [1]. From the executing host point of view, we have the trust on mobile code; this trust is based on the ability of the creator principal to produce good code and on the honesty of the prior

sender principals for not tampering with it or making it malicious; this trust is related to the countermeasures employed in secure mobile agent systems such as the passports for mobile agent credentials and signed security tags for code integrity which can be verified by executing hosts [1], and sand-boxing for host protection [5,1].

Let us now refine these two types of trust using the *Security Enhanced Mobile Agent* (SeA) [1] as an example. We derive six security related trust questions (Q1-Q6) that should be answered at different stages of the life cycle of a typical mobile agent operation.

For a detailed description of the system operation of the SeA refer to [1, 2]. Below we specifically look at the mobile agent operation from trust relation perspective abstracting away the details of the actual operation.

Stage One – Before the deployment of an agent from the owner host: There can be two execution trust questions:

- **Q1**: Before deploying the mobile agents, the agent owner needs to ask if all the foreign hosts on the itinerary list (to be visited) are "trustworthy" in terms of faithfully executing the mobile code (an *execution trust* question).
- **Q1.1**: If yes, then the agent owner needs to determine if all the foreign hosts will have enough trust in the agent owner itself to cooperate with it (that is, a *code trust* question involving *trust symmetry* problem).
- **Q2**: The executing host will not tamper the agent or steal the agent's information, and will migrate the mobile agent to the next host. As will be discussed later this trust is difficult to obtain as it may not be possible to know the trust that the interim executing host has on the remaining hosts. That is, the interim host will need to make a decision on the trustworthiness of the host on the next hop from its point of view which may not be known to the agent owner (an *execution trust* question). Even though having the hosts on the itinerary indicates the initial trust the home base (i.e. the agent owner principal) has on the hosts on the list (including the current host), it does not guarantee that the current (an intermediary) host will consider the next host to be trustworthy. In many cases such trust relationship is often implicitly assumed.

Stage Two – Mobile agent in transit: An execution host may need to ask the mobile *code trust* question as well as the *execution trust* question.

- **Q3**: After receiving a mobile agent and before executing the code, the foreign host may need to be sure that the agent owner host is trustworthy in terms of generating proper mobile code, and every prior executing host is trustworthy in terms of execution and migration. This is thus a hybrid question involving both the *code* trust and *execution* trust.
- **Q4**: The current host needs to determine whether the next host on the itinerary is trustworthy for execution (an *execution* trust question).

Stage Three – Last host: **Q5**: The last host on the itinerary list will need to ask the trust questions as in **Q3** and **Q4**, with the difference that the entity in question is now the agent owner host (and not just the next executing

host on the list), and that the trust on the agent owner is already answered in **Q3** above.

Stage Four – After mobile agent returns to the owner host: Q6: The agent owner needs to check the integrity of the mobile agent and data. The hosts on the propagation list (visited by the agent) may also need to be examined to gain the trust on the results.

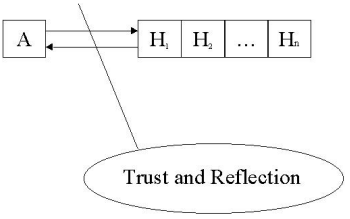

Fig. 1. *Trust* and *reflection* between the agent owner host A and the hosts $H_1 \ldots H_n$ on the itinerary list

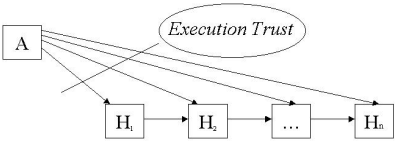

Fig. 2. *Execution trust* the agent owner host A has in the hosts $H_1 \ldots H_n$ and the intermediary execution host has on the next host on the itinerary list

2.1 Trust Symmetry – A Reflection Problem

Let us first focus on **Q1.1**, which is a different type of trust question as opposed to the normal ones such as **Q1**. In this question the agent owner asks whether all foreign hosts will have enough trust in the agent owner to cooperate with it (see Figures 1 and 3). This is a difficult question which relates to *trust symmetry* that may not be known to the agent owner. Hence it can only be answered in a non-deterministic manner, as the agent owner needs to position itself as the other hosts (on the itinerary list) to estimate their trusts on the agent owner from agent owner's point of view, that is, to evaluate the trust symmetry. This is difficult without knowing other agent's mind set (trust parameters). Since in this case the agent owner may not have all the hard facts about other hosts' trust on the agent owner. In this case a trust decision needs to be made with not

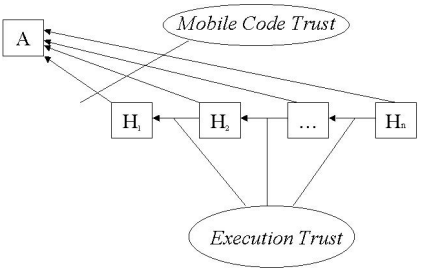

Fig. 3. *Trust reflection* - execution trust on the agent owner host A from the hosts $H_1 \ldots H_n$ and *execution trust* on the previous host from the next host on the itinerary list

only incomplete information and uncertainty in appraising agent owner's trust on other principals (executing hosts), but also taking into account the difficulties in reflecting other principals opinions on oneself (the agent owner). We call this type of trust *reflective trust* and in this paper propose a new approach to capture this in a computationally tractable way. As far as we are aware, such questions involving *trust reflections* in mobile agents have not been raised before.

3 Our Contributions

In this paper, we propose a new method to capture the reflective trust based on Marsh's trust formalism [7]. In particular, our approach provides the following:

1. We formalize the estimating procedure of other agent's trust on the trusting agent itself, thus providing a clear framework to deal with situations where knowledge among interacting agents is incomplete.
2. Our method enables agents to learn from the interactions and dynamically update trust parameters. In additional to Marsh's basic parameter set, the new parameters of trust reflection are included and can be manipulated to aid trust based security decisions.
3. We provide a new trust formalism by including trust reflection, which yields a more general framework than Marsh's. In fact, Marsh's framework can be viewed as a subset of our framework when the trust initiating agents only reason about the trust on other agents without evaluating trust reflection.
4. We apply the new framework to address the mobile agent trust problems raised in Section 2.

Before describing our new formalism and approach, it is useful to briefly review Marsh's trust formalism and notations. We will only mention those elements that are relevant for our discussion. For a full description of the model, refer to [7,8,9].

3.1 Formalizing Trust as a Computational Concept – Marsh's Approach

Marsh presented an approach to formally capture trust as a computationally tractable concept. However, his approach does not provide a clear formalism to deal with the reflection type trust questions that we have raised above. His approach is based on knowledge, utility, situation and experience. Hence this enables dynamic learning through the operating of the system by dynamically re-evaluate the situational trust.

Basic Definitions. The intuition of trust, trust values, trust threshold and their relationships with system knowledge, utility, situation and experience are captured by the following definitions [7].

Definition 1. *An agent (a...z) is an independent entity existing in a world populated with other such entities; each agent is a member of set of agents(A).*

Definition 2. *A situation is a specific point in time relative to a specific agent. Different agents at the same point in time may not consider themselves to be in identical situation.*

Definition 3. *An agent, as a trusting entity, has a basic trust 'value', derived from previous experience. This value, which is dynamically altered in the light of all experiences, is used in the formation of trusting relationships with unknown agents. It is represented as Tx, normalized over (0,1).*

Definition 4. *Trust value is a numerical representation of the fact that x trusts y, as* $T_x(y), x, y \in A$. *The value is normalized over (0,1). This is a trust in the entity only.*

Definition 5. *Situational Trust is the amount of trust in a particular agent in a given situation. It is represented as* $T_x(y, \alpha_x)$, *normalized over (0,1) for x's situational trust in or reliance on y to perform correctly in situation* (α_x). *Stating explicitly trust in entities in a specific situation adds power to the definition of trust by allowing considerations over time in different circumstances.*

Definition 6. *The importance value* $I_x(\alpha_x)$ *is the agent's estimate of how important a situation* (α_x) *is to itself and is normalized over (0,1). This notion is related to the utility of the situation* (α_x).

Definition 7. *The cost of a situation is measured in terms of the problems associated with incompetent or malevolent behaviors on the part of another agent. This is written as* $C_x(\alpha_x)$ *and is normalized over (0,1).*

Definition 8. *The benefit of a situation is the expected benefit of trustworthy behavior from agent(s) being worked with; this notion plays a large part in decisions of whether or not to cooperate in the first place. This is written as* $B_x(\alpha_x)$*, and is normalized over (0,1).*

Definition 9. *The utility of a situation is the expected utility gained by the trusting agent from the trustworthy behavior from agent(s) being worked with; In other words it represents the value and satisfaction gained from from a trustworthy transaction. This is written as* $U_x(\alpha_x)$*, and is normalized over (0,1).*

Remark 1. **Relationships between costs, benefits and agent situational importance:** Importance goes further than weighing of costs and benefits. It may also include some knowledge or assumptions about future benefits and preparation for further cooperation. As argued by Marsh that the importance adds subjective measure for the situation as opposed to the rational measures by utility.

Remark 2. **Trust, experience and situation:** Since trust is based on an agent's experience of previous interactions and situations to a large extent, and is subjective in that it depends on individuals, some method of showing whether or not an agent is known is needed. This is referred to as acquaintance in that an agent becomes acquainted with another. This has been simplified to a Boolean concept of whether an agent knows another or not. The concept of knowledge, or acquaintance is represented as $K_x(y)$.

Here is a summary for the basic trust notations used by Marsh [7]:

- Situations are represented by: $\alpha_x \ldots \omega_x$.
- Individual agents are represented by $a...z$, and are members of A, the set of all agents.
- Basic trust value for agent x: T_x.
- General trust agent x has in agent y: $T_x(y)$.
- Situational trust (Reliance) x has in y in situation α: $T_x(y, \alpha)$.
- Importance of situation α_x to agent x: $I_x(\alpha)$.
- Potential costs to agent x following untrustworthy behavior from another trust agent in situation x: $C_x(\alpha)$.
- Potential benefits to agent x following trustworthy behavior from another trust agent in situation x: $B_x(\alpha)$.
- Representation of whether agent x knows (is acquainted with) agent y: $K_x(y)$.

Remark 3. We will drop the subscript to situation in the rest of this paper, as the situations shall be mutually shared between interacting agents.

Basic Equations. Approximation of the trust theory can be captured by the following equation of situational trust [7]:

$$T_x(y, \alpha) = T_x(y) * U_x(\alpha) * I_x(\alpha) \tag{1}$$

Remark 4. The above may not always be binding as the decision to trust a specific agent may also be related to the competence of the agent in the given situation, as observed or experienced in previous situations. Thus a notion of trust threshold needs to be introduced which allows the competence to play a role in the trust decision making process. Listed below is the cooperation rule developed by Marsh [7]:

Cooperation Rule:

$$If\ T_x(y, \alpha) > Cooperation\_Threshold_x(\alpha) \Rightarrow\ Will\_Cooperate\ (x, y, \alpha) \tag{2}$$

Where:

$$Cooperation\_Threshold_x(\alpha) = \frac{Perceived\_Risk_x(\alpha)}{Perceived\_Competence_x(y, \alpha)} * I_x(\alpha) \tag{3}$$

$$Perceived\_Risk_x(\alpha) = \frac{C_x(\alpha)}{B_x(\alpha)} * I_x(\alpha) \tag{4}$$

$$Perceived\_Competence_x(y, \alpha) = T_x(y, \alpha)\ or\ T_x(y)\ or\ T_x \tag{5}$$

3.2 The New Approach

Going back to the discussion in section 2.1, we basically need to answer the question: "For any given situation, given our own utility preference on this situation (utility, cost, benefit and importance), and our trust on other agents (either in this situation or in general), what is trust the other agents have on us ?"

Obviously, based on Marsh's existing framework, to answer this question we need to have utility preference of other agents; in the worst case where we do not know this, the existing framework will not work.

Hence we need a new approach where the other agent's trust on us can be derived from our own utility preference and trust on other agents. We propose a novel approach where three different views of the trusting agent can be modelled for a given situation, namely *partnership, optimistic and pessimistic views.* In a partnership view, the other agent will treat the current situation in the same way we think about the situation, that is, they consider that in this situation their satisfaction level is the same as ours (utilities). Hence they would trust us equally like we trusted them (the trust value). Based on this intuition, we can formulate procedures using all the base parameters in the initial Marsh model to derive from our point of view other agent's trust in ourselves in a computationally tractable way. In the case of an optimistic view, we say that the other agents will treat the current situation more favorably than ourselves; that is, they would have a high utility value for the situation and higher trust for us. Finally, a pessimistic view is the opposite of the optimistic view.

3.3 Derivation of Other Agent's Trust on Ourselves – A Reflective Approach

Now we can formalize the intuitive concept developed above. First, we introduce some new syntactical notations: we add superscript to Marsh's notations to indicate owner agent's estimate of other agent's utility on the situation and trust on owner agent. Also included in the superscript is the view the owner agent holds when making such estimates. For example, to represent other agent's trust (other agents denoted as x) on the owner agent A in situation α, from A's point of view, one can write: $T_x^{V(A)}(A, \alpha)$, where: $V(A)$ is the view held by the home base when making the estimate. *V(A)* ::= *Partnership* | *Optimistic* | *Pessimistic*. Naturally the owner agent's trust on other agents (x) is represented as: $T_A^{V(A)}(x, \alpha)$. However, for simplicity we drop the superscript in this case, as it conveys no additional meaning when an agent is estimating its own view in this case.

Now we can restate our original question raised in Section 2.1, with the new notations as follows:

Let Home base (ourselves)be A; the set of agent hosts in the network be X; hosts on the mobile agent's itinerary list be $x = H_1 \dots H_n \subseteq X$; and situation be α.

We have the following assumptions:

Trust: $T_A(x, \alpha), T_A(x), T_A$.
Utility: $U_A(\alpha), I_A(\alpha), C_A(\alpha), B_A(\alpha)$.
Decision Functions: $Cooperation\_Threshold_A(\alpha), Will\_Cooperate(A, x, \alpha)$.
Decision Rule: $T_A(x, \alpha) \geq Cooperation\_Threshold_A(\alpha) \Rightarrow Will\_Cooperate(A, x, \alpha)$.

We then need to calculate the following in order to solve the trust *reflection* problem raised in Section 2.1.

Trust: $T_x^{V(A)}(A, \alpha)$, $T_x^{V(A)}(A)$, $T_x^{V(A)}$.
Utility: $U_x^{V(A)}(\alpha)$, $I_x^{V(A)}(y)$, $C_x^{V(A)}(\alpha)$, $B_x^{V(A)}(\alpha)$.
Decision Functions: $Cooperation\_Threshold_x^{V(A)}(\alpha), Will\_Cooperate^{V(A)}(x, A, \alpha)$.
Decision Rule: $T_x^{V(A)}(A, \alpha) \geq Cooperation\_Threshold_x^{V(A)}(\alpha) \Rightarrow Will\_Cooperate^{V(A)}(x, A, \alpha)$.

In order to complete the above trust computation, we need to arrive at some formulae for:

- A's point of view on x's view of utilities, cost and benefit.
- A's point of view on x's view of importance of the situation.
- A's point of view on x's trust on A.

Derivation of Reflective Trust Parameters. We present here three views that the owner agent may hold at any given situation, namely the views of *Partnership, Optimism and Pessimism.*

Case One – Partnership View. On *utility* estimation, one could intuitively argue that if A does not know the view of x (in terms of utility, i.e. cost and benefits) for a given situation, then in this partnership view, A believes that there is no advantage or disadvantage for x (neutral): $C_x^{V(A)=partnership}(\alpha) = B_x^{V(A)=partnership}(\alpha)$. On *importance* estimation in this partnership view, A believes that this situation is as important to x as it is to itself; thus $I_x^{V(A)=partnership}(\alpha) = I_A(\alpha)$. The trust value estimation is independent of the views. In terms of the trust values, we may have four categories of estimation that are applicable to all the views A may have on x's utility and the situation. 1) Since A does not know anything, the minimum case here is that A believes that x will associate the default (generic) trust on A and A use its own trust value as the estimate where a global default is not available. Thus $T_x^{V(A)} = T_A$. 2) A does not know x, but x may know A, so $T_x(A)$ may indeed be possessed by x. But A does not know this. So in answering the question (i.e. A is asking will x cooperate with A. A will still associate the trust as above. 3) In this case, A knows that x knows A and thus:$T_x^{V(A)} = T_x(A)$. 4) Finally there can be a situation where the trust value can be thought as possessed by A already: $T_x^{V(A)}(A, \alpha) = T_x(A, \alpha)$.

Case Two – Optimistic View. On *utility* estimation, in this view, A believes that x believes that the current situation will give a higher satisfaction level than it would give to A, and that the *cost* will be *lower* relative to the $benefit$. Then the following is obtained: $C_x^{V(A)=partnership}(\alpha) < B_x^{V(A)=partnership}(\alpha)$. On *importance* estimation, in this optimistic view, A believes that this situation is more important to x as it is to itself A. Thus $I_x^{V(A)=partnership}(\alpha) > I_A(\alpha)$. On *trust* value estimation, this is view independent and thus it is the same as in *Case One* above.

Case Three – Pessimistic View. On utility estimation, this view considers the case when A believes that x believes that the current situation will give a lower satisfaction level than it would give to A , and that the *cost* will be higher relative to the $benefit$. Then the following can be obtained: $C_x^{V(A)=partnership}(\alpha) < B_x^{V(A)=partnership}(\alpha)$. On *importance* estimation, in this optimistic view, A believes that this situation is less important to x as it is to A; thus $I_x^{V(A)=partnership}(\alpha) < I_A(\alpha)$. On *trust* value estimation, it is view independent and thus is the same as in *Case One* above.

3.4 Algorithm for the Trust Model Update

One needs to note that the reflexive trust is only an estimate, and can be wildly wrong [8]. The main idea here is to provide a starting point for the trustor to estimate the initial trust reflection from the trustee and as more interactions occur this value should converge to the real trust reflection. This could mean that even when the initial estimate of reflexive trust is below the cooperation threshold, the trustee may still cooperate implying that the

trustee has a more positive view than the trustor has estimated initially. Therefore, we need a procedure to update the trust values dynamically in a course of on-going interactions such that the converging trust value can be computed. Let us now consider such a trust update algorithm. We look at both successful and failed interactions. The requirements of the mobile agent trust model is to be able to evaluate the trust reflection to aid low level security decisions. More specifically we need to be able to use our trust value and utility preferences as a starting point to set up some initial estimates for trust reflection to enable interactions in the mobile agent context. We then need to be able to update the trust values based on the observed behaviors from the other agents.

Trust Update Algorithm:

1. Calculate trust on all the potential candidates for cooperation from A's point of view using A's own utility preference and trust knowledge.
2. Select the agents that have passed the cooperation threshold for both trust and trust reflection from A's point of view.
3. For all the selected but unknown agents ($K_x(y)$= false), derive their utility and trust on A from A's point of view by adopting the *partnership* view to enable first time interaction.
4. For an unknown agent, if interaction failed at first time, then it will be captured as an unfriendly/malicous agent and A stops any further interactions. However, if the first interaction is successful then the interaction will continue even when the cross over margin is negative.
5. For a known agent($K_x(y)$= true) , if the interaction is successful then it will be associated with an *optimistic* view for a future interaction. Its utility and trust data are updated as given below:
   - Utility, benefit will be incremented and cost decremented by a set amount.
   - Importance will be incremented by a set amount.
   - Trust value will be incremented by a set amount.
   - Recalculate, risk, competence, situational trust and trust threshold for future interactions.
6. If the interaction is unsuccessful then it will be associated with a *pessimistic* view for a future interaction. Its utility and trust data will be updated as below:
   - Utility and benefit will be decremented and cost incremented by a set amount.
   - Importance will be decremented by a set amount.
   - Trust value will be decremented by a set amount.
   - Recalculate risk, competence, situational trust and trust threshold for future interactions.

4 Application to Mobile Agent Security

Now we can apply the new trust model and the algorithm to mobile agent security problems that we introduced in Section 2.1.

Example 1. Assume that the following is given for the agent owner A and host x, and that A knows nothing about x initially. Here we set out to evaluate the reflexive trust the x has on A using a *partnership view*. We also assume seven successful interactions followed by one (or more) failed interaction(s). We set margin value to be 0.05; the increment delta to be 0.1 and decrement delta to be 0.2. These are intended to simulate the social norm that trust is hard to build but easy to destroy.

Table 1. Initial Parameter Values

Initial Parameters	Values
$K_A(x)$	$false$
$U_A(\alpha)$	0.45
$I_A(\alpha)$	0.60
$B_A(\alpha)$	0.70
$C_A(\alpha)$	0.50
T_A	0.4
$Risk_A(\alpha)$	0.43
$Competence_A(\alpha)$	0.4
$T_A(x, \alpha)$	0.108
$Threshold_A(\alpha)$	0.64
$Margin$	-0.53
$Will\_Cooperate(A, x, \alpha)$	Yes

Margin[1] is used to determine the threshold crossover point to avoid oscillation. Cooperation [2] is enabled for the first time interaction for an unknown host using *partnership view*. The increment/decrement amounts chosen here for various utility and trust parameters are for illustrative purpose only. Refer to [11] for a detailed study on how the amount of increment/decrement and their ratio can impact on the trust evolution. We obtain the simulated results by applying the developed trust derivation rules in Section 3 and the update algorithm in Section 3.4. See Table 2 for the results.

4.1 Remarks on the Simulation Results

The model will have an upper limit of 1 for the parameters that are normalized to 0 to 1, and a lower limit to 0.1 which is the smallest increment or decrement in the system when updating. It can be observed from the simulation that it takes seven successful interactions to bring the trust level to the upper limit for the given condition and only take 1 failed interaction to take agent to below

[1] Cross above if $T_x(A, \alpha) > Will\_Cooperate(x, A, \alpha) + margin$; Cross under if $T_x(A, \alpha) < Will\_Cooperate(x, A, \alpha)$ - $margin$

[2] See algorithm in Section 3.4

Table 2. Trust Evolution in Light of Experience, Partnership = +, Pessimistic = -

Parameters	Init State	1st	2nd	3rd	4th	5th	6th	7th	1st(failure)
$U_x(\alpha)$	0.45	0.55	0.65	0.75	0.85	0.95	1	1	0.65
$I_x(\alpha)$	0.60	0.7	0.8	0.9	1	1	1	1	0.8
$B_x(\alpha)$	0.70	0.8	0.9	1	1	1	1	1	0.65
$C_x(\alpha)$	0.50	0.4	0.3	0.2	0.1	0.1	0.1	0.1	0.8
T_x	0.4	0.5	0.45	0.55	0.65	0.75	0.85	1	0.8
$Risk_x(\alpha)$	0.43	0.35	0.27	0.18	0.1	0.1	0.1	0.1	0.3
$Competence_x(\alpha)$	0.4	0.5	0.45	0.55	0.65	0.75	0.85	1	0.45
$T_x(A, \alpha)$	0.108	0.135	0.234	0.37	0.55	0.71	0.85	1	0.234
$Threshold_x(\alpha)$	0.64	0.47	0.29	0.29	0.15	0.13	0.11	0.1	0.53
$Margin$	-0.53	-0.51	-0.24	0.07	0.39	0.58	0.73	0.9	-0.29
$Will\_Cooperate(x, A, \alpha)$	Yes	Yes	Yes	**Yes**	Yes	Yes	Yes	Yes	**No**
$Views$	+	+	+	+	+	+	+	+	**-**

the cooperation level, thus enable the trust model to reflect closely the norms of trust in general society.

Cooperation crossover margins also follow a similar trend. If required it can play a role in determining the optimum partnerships among a set of selected agents by using the margin as a safety device. However, this aspect is not considered in this example as we mainly focus on the reflection trust relationships the agent owner has with a host or hosts.

5 Concluding Remarks

In this paper, we have considered a trust framework for mobile agent based systems. We have discussed some fundamental trust questions that arise in the various stages of a mobile agent based system operation. Within this context, we have considered trust properties such as execution trust and mobile code trust. Then we introduced the problem of trust reflection in mobile agents and proposed a new formalism to capture and reason about trust reflection in the decision making process. Our framework captured the intuition of reasoning about other agents' view on the owner agent from owner agent's point of view. We have presented three possible views and the utility and trust update algorithm. Finally we applied the new formalism and the developed algorithm to the mobile agent trust questions raised earlier and provided an analysis of the simulated results. We believe that the work presented in this paper provides a starting point for setting up a general framework for reasoning about trust in mobile agent based systems.

Currently we are refining the trust update algorithm to enable the usage of agent authenticity information in addition to the observed evidence for trust [3], thus allow different trust update dynamics for authenticated and non-authenticated agents. We are also implementing the proposed trust model in a secure mobile agent system prototype [2], which would be helpful to compare the real performance results with the simulated ones.

References

1. Varadharajan,V., Security Enhanced Mobile Agents, Proc. of 7th ACM Conference on Computer and Communication Security, 2000.
2. Varadharajan, V. and Foster, D., A Secure Architecture and Demonstration of a Secure Mobile Agent Based Application. In the Proceedings of IASTED International Conference on Networks, Parallel and Distributed Processing and Applications 2002.
3. Lin, C. and Varadharajan,V., On the Design of a New Trust Model for Mobile Agent Security, submitted for publication, 2003.
4. Tan, H. K., and Moreau, L., Trust Relationships in a Mobile Agent System," presented at Fifth IEEE International Conference on Mobile Agents, Atlanta, Georgia, December, 2001.
5. Gong, L., Java Security Architecture (JDK1.2), Draft document, Revision 0.5," available from Java Developer Connection, http://java.sun.com.
6. Chess, D. M., Security Issues in Mobile Code Systems, Mobile Agents and Security, Editor Vigna, LNCS 1419, Springer-Verlag, 1998.
7. Marsh, S., Formalising Trust as a Computational Concept, PhD thesis, University of Stirling, 1994.
8. Marsh, S., Trust in DAI, In Castelfranchi and Werner (ed), Artificial Social Systems, Springer Lecture Notes in Artificial Intellegence, vol. 830, 1994.
9. Marsh, S., Optimism and Pessimism in Trust, Technical Report CSM-117, Department of Computer Science, University of Stirling, 1994.
10. in, C. and Varadharajan, V., On Design of a New Trust Model for Mobile Agent Security, submitted for publication.
11. Yu, B. and Singh, M.P., A Social Mechanism for Reputation Management in Electronic Commerce, In Proceedings of the 4th International Workshop on Cooperative Information Agents, pp. 154–165, 2000.
12. Abdul-Rahman, A. and Hailes, S. A Distributed Trust Model. In Proceedings of the 1997 New Security Paradigms Workshop, pages 48–60. ACM, 1997.
13. Abdul-Rahman, A. and Hailes, S., Supporting Trust in Virtual Communities. In Proceedings of the Hawaii International Conference on System Sciences 33,Maui, Hawaii.
14. Bierman, E. and Cloete, E., Classification of Malicious Host Threats in Mobile Agent Computing. In the proceedings of SAICSIT 2002.
15. Farmer, W.M., Guttman, J.D. and Swarup, V., Security for Mobile Agents: Authentication and State Appraisal. in In Proceedings of the 4th European Symposium on Research in Computer Security, (Berlin, 1996), Springer Verlag, pp 118–130.
16. Gray, R.S., A Flexible and Secure Mobile Agent System, 4th Annual Tcl/Tk Workshop Proc.
17. Jansen, W., Countermeasures for Mobile Agent Security, Comupter Communications, Special Issue on Advances of Network Security.
18. Karnik, N. and Tripathi, A., Security in Ajanta Mobile System. Software Practice and Experience, John Wiley and Sons.
19. Necula, G. Proof-Carrying Code. Proceedings 24th Annual Symposium on Principles of Programming Languages (Paris, France, Jan. 1997),ACM, New York. 106–119.
20. Oppliger, R. Security Issues Related to Mobile Code and Agent-Based Systems. Computer Communications, 22 (12). pp. 1165–1170, July 1999.

21. Sander, T. and Tschudin, C., Towards Mobile Cryptography. in Proc. of the IEEE Symposium on Security and Privacy, USA, (1998), IEEE Computer Society.
22. Tan, H.K. and Moreau, L., Trust Relationships in a Mobile Agent System. in Fifth IEEE International Conference on Mobile Agents, (Atlanta, Georgia, December, 2001), Springer-Verlag.
23. Vigna, G., Protecting Mobile Agents through Tracing, LINCS 1419, 1998.
24. Wilhelm, U.G., Staamann, S. and Buttyán, L., Introducing Trusted Third Parties to the Mobile Agent Paradigm. in Secure Internet Programming: Security Issues for Mobile and Distributed Objects, LNCS 1603, (1999), Springer-Verlag.

An Authorization Model for E-consent Requirement in a Health Care Application

Chun Ruan[1] and Vijay Varadharajan[1,2]

[1] School of Computing and Information Technology
University of Western Sydney, Penrith South DC, NSW 1797 Australia
{chun,vijay,yan}@cit.uws.edu.au
[2] Department of Computing
Macquarie University, North Ryde, NSW 2109 Australia
vijay@ics.mq.edu.au

Abstract. More and more coordination of health care relies on the electronic transmission of confidential information about patients between different health care services. Since the patient data is confidential, patients should be able to delegate, give or withhold e-consent to those who wish to access their electronic health information. Therefore the problem of how to represent and evaluate e-consent becomes quite important in secure health information processing. This paper presents an authorization model for e-consent requirement in a health care application. The model supports well controlled consent delegation, both explicit and implicit consent and denial, individual based or role based consent model, and consent inheritance and exception. A system architecture for e-consent is also presented.

Keywords: Authorization, e-consent, access control, security management

1 Introduction

Computer information processing and electronic communication technologies play an increasingly important role in the area of health care. More and more coordination of health care relies on the electronic transmission of confidential information about patients between different health care and community services. However, since the patient data is confidential, the need for electronic forms of patient consent, referred to as e-consent [3] has to be considered. Patients should be able to delegate, give or withhold 'e-consent' to those who want to access their electronic health information. That is, the secure health information technology needs to support confidential patient and service provider interactions.

The presence of an electronic distributed environment means that patient information will be able to be distributed over the network. More clinical workers in a greater diversity of locations can access it more often and more easily. Consequently, without the existence of some e-consent mechanism, such widespread information could be accessed by unauthorized individuals, or used for purposes not originally consented to by the patient, which can lead to substantial

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 191–205, 2003.

breaches of personal privacy. The main application areas that need e-consent are those that support coordinated health care. This is characterized by data-sharing among multiple teams of health care professionals and institutions, and uses and disclosures of patient data that are additional to those that have hitherto occurred. The aim is to ensure that appropriate patient authorization is obtained in any access to patient information, which would not normally be accessible to a practitioner. In this way, the patients are able to actively participate in the decision making and in the governance of the health services they need.

In an electronic environment, the patient consent is represented and stored in a computer, and the existence of it is determined by automatic processes. For example, a set of computer rules can be defined to determine whether clinical staff working in a hospital might have their right to access electronic patient records. This is actually the so-called access control in computer information security. A number of characteristics of consent are important, including: consent delegation and delegation control; consent inheritance and exception; the explicit and implicit consents; conflict resolution policy when consent and denial exist at the same time for an individual or a specific role; the specificity and boundedness of consent; consent update and revocation. Therefore, a flexible e-consent system is needed which is capable of representing and evaluating a complex set of consent instructions with inclusions and exclusions conditions to access information.

This paper is concerned with e-consent in relation to health data disclosure. While much is known about the access control technology, very little work exists to determine how a patient's consent to view private information is expressed and evaluated in an on-line electronic environment. Based on our previous work [6], we will present a logic based approach to representing and evaluating patient consent to the disclosure of their data, with particular emphasis on security. We will first present an architecture for e-consent system within health care and examine a range of e-consent requirements. We then show how our proposed authorization models can well support these various e-consent requirements.

The rest of this paper is organised as follows. Section 2 discusses access control models and and their application on e-consent. Section 3 presents electronic patient records. Section 4 proposes a system architecture for e-consent system, while Section 5 presents a logic approach to different e-consent requirements. Finally, Section 6 concludes the paper with some remarks.

2 Access Control Models and E-consent

2.1 Discretionary Access Control and Logic Based Access Control

Access control is needed in any secure computer system that provides for controlled sharing of information and other resources among multiple users. It comprises all system mechanisms that are required to check whether an access request issued by a particular user is allowed or not, and moreover all mechanisms that are required to enforce the decision accordingly. Access control models, or authorization models, provide a formalism and framework for specifying, analyzing and evaluating security policies that determine how access is granted to

and delegated among particular users. Various access control models have been published [2]. Some are defined in terms of well known abstraction of subjects, objects and access rights and some in terms of roles, permissions and users. Some models are formulated in matrix and rules, some in graphs, while other in logic.

Discretionary Access Control model has long been a widely accepted and used model in the real world secure systems. It govern the access of subjects to the objects on the basis of the subject's identity and of authorizations that specify, for each subject and object in the system, the access rights (e.g., read, write) the subject is allowed on the object. Objects are the passive entities storing information, such as patient health data files, records, fields in records, etc. Subjects are active entities that access the objects, such as doctors, nurses, general practitioners, etc. Each request of a user to access an object is checked against the specified authorizations. If there exists an authorization stating that the user can access the object in the specific access right, the access is granted, otherwise it is denied. Discretionary access control represents a flexible way to enforce different protection requirements, especially cooperative yet autonomous requirements. In [9], Varadharajan et al developed an access control model for mental health application.

On the other hand, *logic based* approaches have been developed by many researchers recently for the purpose of formalizing authorization specifications and evaluations [1,4,5]. The advantage of this methodology is to separate policies from implementation mechanisms, give policies precise semantics, and provide a unified framework that can support multiple policies. Logic based method usually has strong expressive and reasoning power which is particularly useful in supporting a complex set of e-consent instructions with nested inclusions and exclusions conditions to access health data.

2.2 Role-Based Access Control

Another paradigm of access control models is *Role Based Access Control*(RBAC) [7]. The central notion of RBAC is that accesses are associated with roles and users are assigned to appropriate roles thereby acquiring accesses.

Classic access control is based on the individual (subject) accessing a resource (object).

subjects $\rightarrow$ objects

In many situations, privileges are associated with posts other than individuals. Individuals get their privileges because their posts or positions in the organization. In other words, whoever get the post would get the privileges of the post. When people leave the organization or change the positions, their privileges will be revoked or changed, too. This happens in many organizations from the viewpoint of organization administration. For example, a doctor in a hospital can access the patients' information in the hospital. If the doctor leaves the hospital, he/she usually lose the capability to access the patients' information, too. If the number of subjects and objects is large, individual access control becomes

difficult. Each individual needs to be assigned each access right when they get a position in the organization and revoked each access right if the person changes the post or leaves the organization. When privileges are indeed assigned to roles other than individual subjects, role-based access control can greatly simplify the administration work.

In role-based access control, roles are placed between the user and the resource and subjects get their access rights indirectly by assigning access rights to roles and roles to subjects. Roles describe rights, duties and tasks that people have to perform. When people leave or change roles, only the mapping from subjects to roles need to be revoked or changed. On the other hand, if the duties of the roles change, only the mapping from roles to objects need to be changed. Roles provide a more abstract viewpoint on access control.

subjects → roles → objects

The concept of role also applies to the provision of patient data in health care contexts. Some consents may be given by patients in relation to roles ("yes, I consent to have a pathology test done on those samples"). Multiple individuals may perform particular roles at different times, e.g. because of the need for shift-work in intensive-care.

Roles can be organized into hierarchies so that consents can be inherited, which could greatly reduce the amount of explicit consent specification. Roles can also be delegated. One entity may act on behalf of another. Attorney is an example to this relationship. Other examples include guardians of minors and of people who are not psychologically capable of managing their own affairs. Roles are created according to the job functions in an organization and users are assigned roles based on their qualifications, experience or responsibilities. In [8], Thomsen presented a role-based application design and enforcement in a health care environment.

3 Electronic Patient Records

The patient record is an account of a patient's health and disease after he or she has sought medical help. The record contains findings, considerations, test results and treatment information related to the disease process. The development of an electronic patient record comes from the increasing demand for well-structured and accessible patient data on one hand, and fast developments in computer technology on the other hand. Computer technology has the advantage to improve legibility, accessibility, and structured information management. Electronic patient records may be available to the clinician at any point where electronic access is provided to the records, and they allow simultaneous access by several clinicians at different sites, whereas paper records have to be physically present at the point of use. Data retrieval and update of electronic patient record is easier than from paper, since electronic records are physically more accessible to their users than paper records. It is also more convenient to inte-

grate the content of electronic records for audit and analysis purposes. Efficient information management is critical to the practice of medicine. The existence of an up-to-date, complete database of the medical records and dependable patient data not only enhances the quality of medical care but also improves the teaching and continuing medical education and research.

The following is an example of electronic patient record format.

A . Essential Personal and Contact Details
- A1 . Name—surname, first name
- A2 . Address—street, suburb, state, postcode
- A3 . Date of birth
- A4 . Phone number—home and/or work and/or mobile
- A5 . Payment method—private fee, bulk billing, or 3rd party fee

B . Optional Personal and Contact Details
- B1 . Title (Miss, Ms, Mrs, Mr or Dr)
- B2 . Alias or preferred name
- B3 . Email address
- B4 . Fax number
- B5 . Occupation
- B6 . Gender (male, female or unknown)
- B7 . Marital status (divorced, married,...)
- B8 . Ethnicity (African, Aboriginal, etc)
- B9 . Country of birth
- B10 . Next of kin—name, address and phone number
- B11 . Employer—name, address and phone number
- B12 . Family members—name and address

C . Clinical Related Personal Details
- C1 . Status (regular, casual, transferred or visitor patient)
- C2 . Provider—name of doctor examining the patient
- C3 . Location of provider—(medical center, specialist center, ...)
- C4 . Precedures/treatment code (for billing purposes)
- C5 . Pathology and radiology results(in/out)—in or not in
- C6 . Visit history—all visits, to this location of provider
- C7 . Next appointment—name of preferred doctor, date and time

D . Health Details
- D1 . Consultation
  - D11 . Date of consultation
  - D12 . Current health problems and complains
  - D13 . Examination and notes taking
  - D14 . Management plan
  - D15 . Pathology and radiology requests
  - D16 . Referrals
  - D17 . Follow up and recall arrangements
- D2 . Medical History (all consultation information)
- D3 . Sensitive Medical History
  - D31 . STD(HIV/AIDS) (all consultation information)

D32 . Gynaecological Conditions (all consultation information)
D4 . Other History
D41 . Smoking and alcohol history
D42 . Employment details
D43 . Operation history
D44 . Family history
D5 . Allergies and Sensitivities
D6 . Immunization Record
D7 . Medication History
D8 . Pathology and Radiology Results

4 A System Architecture for E-consent

In this section, we present a system architecture that enables various requirements for e-consent to be developed.

Dimensions of Consent

Consents may involve subjects to whom the consents are given, information (data) to be protected, access rights allowed or prohibited on the information, and subjects who issue the consent.

Subjects: Roles and Individuals

In the context of e-consent for health care, the consent may be assigned on the basis of an individual's identity or a clinical role within an organization. Some consents may be given by patients in relation to identified individuals, e.g., "yes, I consent to send those details to Dr Smith". In other circumstances, the consent is for a role, such as, "yes, I consent to have a pathology test done on those samples". Roles can be organised into different hierarchies so that the consent can be inherited. The supervision role hierarchy, for example, is based on the organization management hierarchy; the isa role hierarchy is based on generalization; and the activity hierarchy is based on aggregation. For example, Figure 1 is a possible role hierarchy based on aggregation.

Data

To allow consent inheritance along the data dimension, data could be organised into hierarchies. For example, the electronic patient record given in Section 3 can be organised into the the hierarchy shown in Figure 2. A consent to read health details (D) means a consent to read all data associated with the episodes of care (D1,D2,D3 and D4), and all the data captured in various events of care.(D5, D6, D7 and D8).

Access Rights

Usual access rights such as read, write, and update apply to the patient data. Access rights can also be organised into hierarchies to allow inheritance along

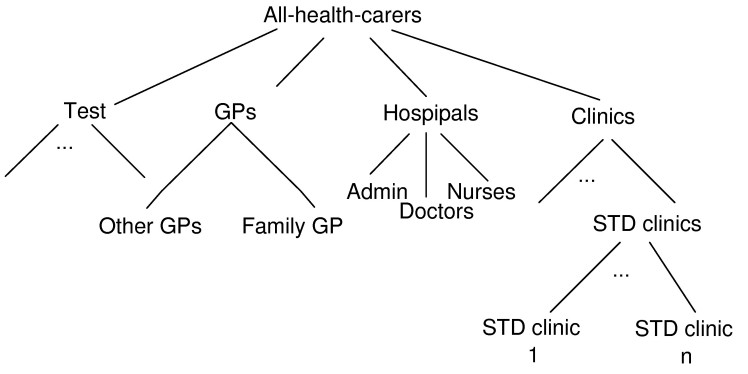

Fig. 1. A role hierachy.

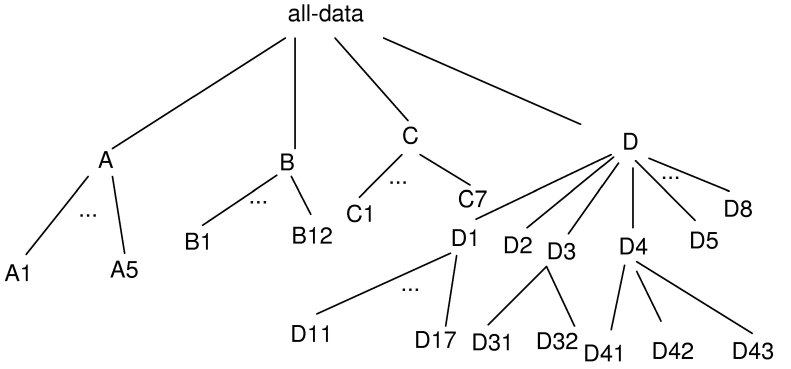

Fig. 2. An object hierachy.

this dimension. A consent to updating for example, may also imply a consent to reading and writing.

Consent and Denial

Both consents and denials are needed in a flexible e-consent system. Denials are useful when patients want to express explicitly that some disclosure is forbidden. In some e-consent models where the presumption is consent, the explicit denial will override the default consent. For example, some countries establish default consent in relation to organ donation that are overridden by an explicit denial. In other e-consent models where denials are established as default, the explicit denial could prevent further assigning of consent accidentally or intentionally.

Role Delegation and Consent Delegation

In some circumstances, a patient may wish to delegate the capability to grant consent to nominated representatives or medical practitioners, who may further wish to delegate the power to consent to other health professionals. This is usually done for flexibility, cooperation, and convenience of the carer. Note that the patient still keeps the capability to issue the consent after the delegation.

In role delegation, a patient delegates all of his/her privileges of consent to the nominated entity. However, in some situations, a patient may only wish to partially delegate his/her capability of creating consent to others, e.g., delegate the consent on the information regrading the treatment details but not the personal details.

Conflict Resolution

Because of role delegation and consent delegation, multiple grantors may exist for a specific consent and hence conflicts may arise. For example, a patient may wish to deny all information relating to HIV to be open to his/her immediate family, but his/her family GP, to whom he/she has delegated the privilege of granting consent, may want to disclose the information to the patient's immediate family. In this case, the patient's immediate family may receive two conflicting authorizations, consent and denial. A proper conflict resolution policy is then needed.

Control of Delegation

As we said before, to achieve high quality of treatment, a patient may wish to give the carer more flexibility by delegating them the required rights. On the other hand, to protect his/her privacy, a patient may not wish to lose control on his/her health data. One solution to this problem is to give a patient higher priority than his/her delegate, so that once conflict occurs, the patient's consent grant will override the other's. In addition, proper constraints on delegation are needed to avoid undesirable situations. For example, a doctor receiving the right to grant for a patient's health data should not be able to deny the patient to read his/her own health data by issuing the patient a negative authorization.

Consent Inheritance and Exceptions

As in many information systems, allowing consent inheritance would greatly reduce the amount of consents that need to be explicitly specified. When consents can be inherited, it is important to support exceptions to avoid undesire inheritance. For example, a consent to the health care professional means a consent to every health carer. A consent to the health care professional followed by a denial to a particular GP means that the GP could not be exposed to the patient's information.

Consent Capability

Usually a patient has the capability to grant consent on his/her own health data. However, in some circumstances a person is physically or legally incapable of giving consent; for instance children or minors, persons in coma, seriously incapacitated or frail people. These cases are usually subject to a variety of health-specific laws, e.g. laws relating to guardianship.

Purposes and Contexts

Sometimes, a consent is assigned on the basis of specific use of information. Common purposes include treatment, cooperation, training, teaching, notification (requests by persons closely associated with the person concerned, such as guardians, partners and immediate family), research, and getting advice from specialists.

Sometimes, consent is assigned based on the current context. A doctor may not be allowed to read the patient's health data in a normal situation, but may be allowed to do so in an emergency situation.

Implicit Consents and Inference Rules

The inherited consents belong to implicit consents. In general, rule based consent specification allows for implicit authorizations to be derived from the explicit authorization set through reasoning. Hence this can greatly reduce the size of explicit authorization set. A system architecture for e-consent is shown in Figure 3.

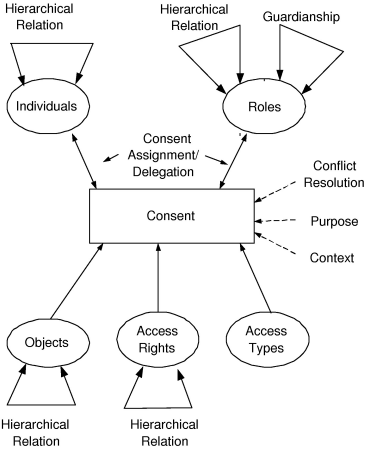

Fig. 3. A System Architecture for e-Consent.

5 Supporting E-consent Requirement

In this section, we show how to use the Delegatable Authorization Program (DAP) presented in [6] to support e-consent requirement.

5.1 DAP

Recall that our language $\mathcal{L}$ is a many-sorted first order language, with four disjoint *sorts* for subject, object, access right and authorization type respectively. In the constant set of authorization types $T = \{-, +, *\}$, $-$ means *negative*, $+$ means *positive*, and $*$ means *delegatable*. A negative authorization specifies the access that must be forbidden, while a positive authorization specifies the access that must be granted. A delegatable authorization specifies the access that must be delegated as well as granted. That is, $*$ means $+$ together with administrative privilege on the access. The partial orders $<_S, <_O, <_A$ represent inheritance hierarchies of subjects, objects and access rights respectively. We suppose $\sharp$ in S denotes the security administrator, and it is not comparable to any subjects in S w.r.t. $<_S$.

The predicate set consists of a set of ordinary predicates defined by users, and one built-in predicate symbol for delegatable authorization, *grant*. *grant* is a 5-term predicate symbol with type $S \times O \times T \times A \times S$. The first argument is the *grantee*, the second argument is the *object*, the third argument is the *authorization type*, the fourth argument is the *access right*, and the fifth argument is the *grantor* of this authorization. Intuitively, $grant(s, o, t, a, g)$ means s is granted by g the access right a on object o with authorization type t.

A *term* is either a variable or a constant. Note that we prohibit function symbols in our language. An *atom* is a construct of the form $p(t_1, ..., t_n)$, where p is a predicate of arity n in P and $t_1, ..., t_n$ are terms. A *literal* is either an atom p or the negation of the atom $\neg p$, where the negation sign $\neg$ represents classical negation. A *rule* r is a statement of the form:

$b_0 \leftarrow b_1, ..., b_k, not\, b_{k+1}, ..., not\, b_m, m >= 0$

where $b_0, b_1, ..., b_m$ are literals, and *not* is the negation as failure symbol. The b_0 is the *head* of r, while the conjunction of $b_1, ..., b_k, not\, b_{k+1}, ..., not\, b_m$ is the *body* of r. Obviously, the body of r could be empty. A *Delegatable Authorization Program*, DAP, consists of a finite set of rules.

Our examples in this section are mainly based on the electronic patient record given in Section 3, its hierarchical structure in Figure 2 and subject hierarchical structure in Figure 1.

5.2 Specifying and Evaluating E-consent Requirement

1. Consent and Denial

We use the authorization predicate *grant* to define consent and denial in the context of health care.

Definition 1. *A subject(individual/role) s consents (denies) another subject s' to exercise access right a on patient data o, which is defined by* $grant(s', o, +, a, p, s)$ $(grant(s', o, -, a, p, s))$.

For example, $grant(Bob, health\text{-}data, +/-, read, Alice)$ means that Alice consents/denies Bob to read her health-data.

By using DAP, it is easy to express the presumed consent requirement. For example, if a patient Bob wants to express that the access to his health data is presumed for anybody unless a explicit denial exists, he can use the following rule.

$grant(\_s, all\text{-}data, +, \_a, Bob) \leftarrow not\ grant(\_s, all\text{-}data, -, \_a, Bob)$

Intuitively, the rule says, $grant(\_s, all\text{-}data, +, \_a, Bob)$ is true if $grant(\_s, all\text{-}data, -, \_a, Bob)$ is not known to be true. Obviously, this consent rule is in favor of the convenience of accessing the health data.

It is also easy to express the general consent requirement. Suppose Bob wants to give a general consent to all the health carers for reading his health data, which can be represented by the following rule.

$grant(all\text{-}health\text{-}carer, all\text{-}data, +, read, Bob) \leftarrow$

If Bob only wish to disclose his health details (D) to all health carers for reading, then the following rule can express this.

$grant(all\text{-}health\text{-}carer, D, +, read, Bob) \leftarrow$

If Bob wants to give a general denial to all the health carers for all the access rights on his data, he can use the following rule to express.

$grant(all\text{-}health\text{-}carer, all\text{-}data, -, \_a, Bob) \leftarrow$

The following rule expresses that Bob give a general denial to a doctor John for all the access rights on his data.

$grant(John, all\text{-}data, -, \_a, Bob) \leftarrow$

2. Consent Delegation and Role Delegation

We first use authorization predicate *grant* to define consent delegation.

Definition 2. *A subject(individual/role) s delegates another subject s' the privilege to grant consent for access right a on patient data o, which is defined by* $grant(s', o, *, a, s)$.

For example, $grant(familyGP, health\text{-}data, *, read, Bob)$ means that Bob delegates his family GP the right to further disclose his health data for reading. Please note that, in our formulation, $*$ means $+$ plus the right to grant. This

means, in this example, the family GP can read the data as well as disclose the data.

The role delegation from $\_s$ to $\_s'$ can be expressed by the following rule, which means that $\_s$ will delegate any of his/her rights on any object to $\_s'$.

$$grant(\_s', \_o, *, \_a, \_s) \leftarrow grant(\_s, \_o, *, \_a, \_g)$$

It is also easy to express the presumed delegation requirement. For example, if Bob wishes to delegate the administrative privilege on his health data to any body unless the explicit denial or consent exists, then the following rule can express this.

$$grant(\_s, all\text{-}data, *, \_a, Bob) \leftarrow not\, grant(\_s, all\text{-}data, -, \_a, Bob), \\ not\, grant(\_s, all\text{-}data, +, \_a, Bob)$$

Intuitively, the rule says, $grant(\_s, all\text{-}data, *, \_a, Bob)$ is true if $grant(\_s, all\text{-}data, -, \_a, Bob)$ and $grant(\_s, all\text{-}data, +, \_a, Bob)$ are not known to be true (negation as failure). Please note that $grant(\_s, all\text{-}data, +, \_a, Bob)$ means a subject can exercise $\_a$ on all-data, but cannot further grant $\_a$ to other subjects.

If Bob wishes to deny a doctor John to access his data unless a explicit consent or delegation is granted, the following rule can express this.

$$grant(John, all\text{-}data, -, \_a, Bob) \leftarrow not\, grant(John, all\text{-}data, +, \_a, Bob), \\ not\, grant(John, all\text{-}data, *, \_a, Bob)$$

3. Capability to Give Consent

To denote the capability to give consent, we define a guardian relation first. We introduce a new predicate, $guardian(s, s')$ with type $S \times S$, which means that s is a guardian of s'. For usual patients, they are their own guardians. For patients who are physically or legally incapable of giving consent, their guardians are different persons (subject to law). Let $own(s, o)$, with type $S \times O$, represent that s is the owner of the data o. Patients are considered to be the owners of their own health data.

Next, let the system administrator $\sharp$ delegate all the access rights on health data to the guardians of the patients.

$$grant(\_s, \_o, *, \_a, \sharp) \leftarrow own(\_s', \_o), guardian(\_s, \_s')$$

Hence, at the beginning, only guardians of patients can give consent to patients' data. However, through consent delegation or role delegation, other persons may receive capability to give consent, too.

4. Conflict Resolution

Let us have a look again at the conflict resolution policy proposed in [6] and see how it works here. First we solve the conflicts in terms of the delegation relations

and give higher priorities to predecessors. In the e-consent context, a patient certainly wish to hold higher priorities than his/her delegates, so that whenever his/her consent instructions are conflicting with other's, his/her instructions will win. This is true for other delegators, too. In fact, to achieve high quality of treatment, a patient may wish to give the carer as much flexibility as possible. On the other hand, to protect his/her privacy, a patient may not wish to lose control on his/her health data. Giving a patient or, in general, a delegator higher priority than his/her delegate would best suit this situation. For example, a patient Bob may wish to deny all information relating to HIV to be open to his immediate family, but his family GP, to whom he has delegated the privilege of granting consent, may want to disclose the information to his immediate family. In this case, the patient's authorization will override his family GP's and therefore his immediate family could not access his HIV related information.

When two conflicting authorizations have the same grantors, the hierarchies of subjects, objects, and access rights are considered and the more-specific take precedence principle will be used. This can support the exceptions in consent inheritance. For example, a patient provides a general consent to a health care professional on the health data, but specifically precludes the disclosure of information about an STD condition, gynaecological procedure; or disclosure to their family GP. By using the more specific-take- precedence policy, the general consent will be overridden by the more specific denials.

If all the above policies don't apply, a patient can simply select denial-take-precedence based method, which is in favor of privacy; or consent-take-precedence based method, which in favor of treatment convenience.

Example 1. For more complex model where qualifications are nested, see the following example based on [3]. A patient Alice delegates to all information to all health carers; but within that denies all information relating to HIV (D3) and consents to information relating to HIV to STD clinics; and finally denies all information to a specific STD clinic-1 (where her mother works). The following rules plus our conflict resolution policy can achieve these requirements.

(1) $grant(\textit{all-health-carer}, \textit{all-data}, *, \_a, \textit{Alice}) \leftarrow$

(2) $grant(\textit{all-health-carer}, D3, -, \_a, \textit{Alice}) \leftarrow$

(3) $grant(\textit{STD-clinics}, D3, +, \_a, \textit{Alice}) \leftarrow$

(4) $grant(\textit{STD-clinic}\text{-}1, D3, -, \_a, \textit{Alice}) \leftarrow$

Let us have a closer look on this. The first rule is a general rule which will propagate downward the subject and object hierarchies defined by Figure 1 and Figure 2, since consent inheritance in both dimensions is supported. therefore all health carers can access D3 which conflicts with the second rule. As Alice is the grantor of all the rules, the more specific-take-precedence principle will be used here; hence the inherited consent instructions from the general rules will be

overridden by the more specific consent rules. This means that the second rule will override the first rule(on D3). For the same reason, the fourth rule overrides the third which again overrides the second.

On the other hand, since Alice is the owner, her instructions can not be overridden by any other person's instructions. Suppose a doctor John issues the following rule:

(5) $grant(STD\text{-}clinic\text{-}1, D3, +, \_a, John) \leftarrow$

This rule conflicts with the fourth rule and will be overridden by the fourth rule since John receives his delegatble privilege on Alice's health data from Alice (through the propagation of rule (1) along the subject hierarchy). Therefore the above four rules meet Alice's requirements.

5. Delegation Control

Cyclic authorizations are prohibited in DAP, as they usually do not make much sense and may cause undesirable situation. Consider for example a patient delegating the "read" right on his/her health data to his/her family GP. It is thus meaningless that the GP grants back to the patient to read his/her data. Moreover, it is undesirable if the GP could deny the patient to read his/her own data. On the other hand, as we mentioned before, giving predecessors higher priorities than successors provides users further control on consent delegation.

6. Purposes and Contexts

We extend the language a little by adding another sort *other* to the language, which contains constants and variables other than subjects, objects, access rights and authorization types. For example, we can put different purposes and contexts in this sort.

We further introduce two predicates, *purpose* and *context*; they both with one argument in sort *other*. The following rule means that a patient Alice consents to all health professionals to exercise any access on her health data if their purposes are for *treatment*.

$$grant(all\text{-}health\text{-}carer, all\text{-}data, *, \_a, Alice) \leftarrow purpose(treatment)$$

Similarly, the following rule states that a patient Bob gives all health professionals the consent for any access to his health data if it is in an *emergency* situation.

$$grant(all\text{-}health\text{-}carer, all\text{-}data, *, \_a, Bob) \leftarrow context(emergency)$$

6 Conclusion

In this paper, we presented an authorization model for representing and evaluating e-consent requirement in a health care application. This model supports well controlled consent delegations, both explicit and implicit consents and denials, individual based or role based consent instructions, and consent inheritance and exception. It is shown that the Delegatable Authorization Program and its conflict resolution policy provide users a good framework to express complex e-consent requirements. In addition, an electronic patient record is discussed and a flexible system architecture for e-consent is also presented.

References

1. E. Bertino, F.buccafurri, E.Ferrari, and P.Rullo, A logical framework for reasoning on data access control policies. In *Proc. of the 12th IEEE Computer Society Foundations Workshop*, pp 175–189, IEEE Computer Society Press, 1999.
2. S. Castano, M. Fugini, G. Martella, and P. Samarati, *Database Security.* Addison-Wesley Publishing Company, 1995.
3. E. Coiera, Consumer consent in electronic health data exchange. *Report*, the University of New South Wales, Australia, 2001.
4. J. Crampton, G. Loizou, and G. O'Shea, A logic of access control. *The Computer Journal*, 44:54–66, 2001.
5. S. Jajodia, P. Samarati, and V.S. Subrahmanian, A logical language for expressing authorizations. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp 31–42, 1997.
6. C. Ruan,V. Varadharajan, and Y.Zhang, Logic-based reasoning on delegatable authorizations. In *Proc. of the 13th International Symposium on Methodologies for Intelligent Systems* pp 185–193, 2002.
7. R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and Charles E. Youman. Role based access control models. *IEEE Computer*, 29(2):38–47, February 1996.
8. D.J. Thomsen, Role-Based Application Design and Enforcement.*Database Security,IV: Status and Prospects*, Elsevier Science Publisher B.V., pp 151–169 1991.
9. V. Varadharajan and C. Calvelli, An access control model and its use in representing mental health application access policy. *IEEE Transaction on Knowledge and Data Engineering*, 8(1):81–95, 1996.

PLI: A New Framework to Protect Digital Content for P2P Networks

Guofei Gu[1], Bin B. Zhu[2], Shipeng Li[2], and Shiyong Zhang[1]

[1] Dept. of Computing & Info. Tech., Fudan Univ., Shanghai 200433, China
gu_guofei@yahoo.com, szhang@fudan.edu.cn
[2] Microsoft Research Asia, Beijing 100080, China
{binzhu,spli}@microsoft.com

Abstract. In this paper, we first propose a novel Public License Infrastructure (PLI) that uses cryptographic threshold secret sharing schemes to provide decentralized public license services for the Digital Rights Management (DRM). This distributed PLI replaces the centralized license server in a conventional DRM system. PLI offers many advantages such as intrusion and fault tolerance, flexibility, scalability, reliability, high availability. We then propose a PLI-based DRM system to provide content protection and digital rights management for users of Peer-to-Peer (P2P) networks. This DRM system is especially useful for small content providers such as peers in a P2P network who cannot afford the conventional server/client based DRM system and traditional distribution channels.

Keywords: Peer-to-Peer (P2P), digital content protection, Digital Rights Management (DRM), Public License Infrastructure (PLI), License Authority (LA), intrusion tolerance, secret sharing, proactive shares update, distributed trust model.

1 Introduction

Wide availability of digital content such as software and digital media and easy access through Internet have prompted the demand for technologies to protect digital content from illegal access or copy. Digital Rights Management (DRM) is a system which manages all rights for digital content from creation to consumption [1][2]. Most DRM systems such as Adobe's EBooks [3], Intertrust's [4], and Microsoft's [5] DRM systems are based on encryption. Digital content is encrypted and distributed. A consumer who wants to play the protected content will first get the access permission and the decryption key. The DRM system enforces the proper usage of the digital content. Typically the access rules or rights and the decryption key are contained in a encrypted license. In these systems, license acquisition is based on the classical server/client model in which a user acquires a license from a centralized and dedicated license server.

This work was done when Guofei Gu was a visiting student at the Microsoft Research Asia.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 206–216, 2003.

Peer-to-peer (P2P) networks have recently attracted increasing attention in both academia and the industry. P2P networks offer many desirable features: adaptation, self-organization, load-balance, fault-tolerance, low cost, high availability, scalability, and a large pool of resources. P2P networks have emerged as a popular way to share huge amounts of data (e.g., [6][7]). Most P2P networks do not have any digital rights management or access control. P2P networks are often blamed for illegally sharing copyrighted materials.

In a conventional DRM system, all license acquisition requests are processed by a centralized license server. This makes the license server heavy-loaded, complex and expensive to run and maintain, and a weak link in the system. Failure of the license server disrupts normal DRM services. On the other hand, small content providers such as a peer in a P2P network may not afford the cost of services of the license server. In this paper, we leverage the P2P concept to propose a novel distributed infrastructure called the *Public License Infrastructure* (PLI) to provide decentralized license services for DRM. PLI consists of many trusted *License Authorities* (LAs) which collectively provide DRM license services for consumers. Secrets are shared among LAs with a threshold secret sharing scheme. PLI provides an inexpensive, reliable, and highly available alternative to the centralized license server in a conventional DRM system. Based on PLI and LAs, we then propose a DRM system which provides content protection and digital rights management for users of P2P networks. The proposed DRM system is especially useful for small content providers such as peers in a P2P network who cannot afford the conventional server/client based DRM system and traditional distribution channels. The proposed DRM system can be considered as a hybrid P2P network where some trusted nodes, i.e., LAs, play a role as mini-servers. To the best of our knowledge, our proposed system is the first distributed DRM license service system and the first DRM system designed for P2P networks.

The paper is organized as follows: Section 2 introduces the background and related work for the paper, which includes the conventional DRM system, P2P networks, and distributed certificate authority schemes. Section 3 presents the proposed public license infrastructure and the DRM system designed for P2P networks. It is followed with discussions in Section 4. We conclude the paper in Section 5.

2 Background

2.1 Conventional Digital Rights Management (DRM) Systems

Conventional DRM systems are based on the server/client model. A license server is used to provide license services for consumers. A typical example is the Microsoft Windows Media Rights Manager [5] which contains the following five processes: packaging, distribution, establishing a license server, license acquisition, and playing the content [8]. These processes are briefly described in the following:

i. **Packaging.** The rights manager encrypts the digital media and then packages the content into a digital media file. The decryption key is stored in an encrypted license which is distributed separately from the media file. Other

information such as a link to the license is added to the media file to facilitate license acquisition.

ii. **Distribution.** The packaged file is distributed to users through some distribution channels such as downloading, streaming, and CD/DVD. There is no restriction on distribution of the packaged content.

iii. **Establishing a license server.** The content provider (referred to as the publisher in the following) chooses a license clearing house that stores the specific rights or rules of the license and runs a license server which is used to authenticate the consumer's request for a license. Licenses and protected media files are distributed and stored separately to make it easier to manage the entire system.

iv. **License acquisition.** To play the protected content, a consumer first acquires a license which contains the decryption key and the rights the consumer has with the content. This process can be done in a transparent way to the consumer or with minimal involvement of the consumer (such as when payment or information is required).

v. **Playing the content.** A player that supports the DRM system is needed to play the protected content. The DRM system ensures that the content is consumed according to the rights or rules included inside the license. Licenses can have different rights, such as start times and dates, duration, and counted operations. Licenses, however, are typically not transferable. Each consumer has to acquire his or her own license to play the protected content.

2.2 Peer-to-Peer (P2P) Networks

With the introduction of the first P2P network several years ago, we have seen an explosive growth of P2P networks and their applications. Well-known P2P networks include Napster [9], Gnutella [10], JXTA [11], Freenet [12], Chord [13], CAN [14], Pastry [15], Tapestry [16], etc. P2P networks are also actively studied and developed by researchers in the fields.

P2P networks offer many desirable features such as redundancy and fault tolerance. Data gradually spreads and gets replicated at many nodes, and thus is highly redundant in a P2P network. High redundancy means high reliability and availability. This can effectively reduce the operational cost and serve more users. Despite these desirable features, P2P networks still lag behind the traditional server/client paradigm in security, efficiency, performance guarantees like atomicity and transactional semantics. P2P networks lack of content rights management and access control which threatens their healthy development and wide adoption. Companies building P2P software have been sued in courts by large content providers for illegal sharing of copyrighted materials the P2P software enables. The proposed DRM system in this paper provides a DRM system for P2P networks.

2.3 Threshold-Based Secret Sharing and Distributed Certificate Authorities

Threshold-based secret sharing [17][18] and proactive secret share updates [19][20] have been studied actively in cryptography. They are the basis for the proposed

system in this paper, as well as many other proposed schemes of distributed certificate authorities. The Intrusion Tolerance via Threshold Cryptography (ITTC) project at Stanford [21] enables a private RSA key to be shared among *k* servers such that any *k-1* or *k-2* of them can decrypt incoming messages without reconstructing the key. The Partially Distributed Certificate Authority (PDCA), an ad-hoc key management scheme proposed in [22], uses a (*k, m*) threshold scheme to distribute services of the certificate authority to a set of specialized server nodes. Each of these nodes is capable of generating a partial certificate using its share of the certificate signing key. A combination of *k* partial certificates can generate a valid certificate. The partial certificates are fixed, or more precisely, logically centralized. A specialized server is needed to combine the partial certificates. The Fully Distributed Certificate Authority (FDCA), another ad-hoc key management solution proposed in [23] and analyzed in [24][25], uses a (*k, m*) threshold scheme to distribute an RSA certificate signing key to all nodes in a network. It also uses verifiable and proactive secret sharing mechanisms to protect against denial of service attacks and attacks to compromise the certificate signing key. Unlike PDCA, FDCA distributes the service to all the nodes when they join the network. There is no need to elect or choose any specialized server nodes. This scheme, however, is based on the assumption that every node has a reliable intrusion and misbehavior detection function, which may be impractical in many applications.

3 Public License Infrastructure (PLI) and PLI-Based DRM

3.1 Public License Infrastructure (PLI) and License Authority (LA)

In a conventional server/client based DRM system, every publisher has to choose a license clearing house with a centralized license server to provide license services for consumers. Small publishers may not afford to do so and, as a consequence, their content will not be protected by the conventional DRM. This is especially true in a P2P network where every user can be both a consumer and a publisher. An inexpensive license service and DRM system is desirable to protect small publishers. The Public License Infrastructure (PLI) we propose here serves as an inexpensive license service provider. PLI is similar to the Public Key Infrastructure (PKI). Instead of signing certificates for public keys, PLI is a decentralized system to provide public license services for all the users in a DRM system. Like Certification Authorities (CAs) in PKI, we have many License Authorities (LAs) in PLI. These LAs are fully trusted in the system. To build PLI for a P2P network, we need to have enough trusted nodes as LAs. In this paper we assume that the underlying network always has enough number of trusted nodes, even when the underlying network is a P2P network.

To build distributed license services, we split a secret into *m* partial secrets and upload the partial secrets to LAs. A (*k, m*) threshold secret sharing scheme is used to split and recover the original secret. A LA can have more than one partial secret. In the extreme case, one LA can contain all the *m* partial shares. In this case, the LA behaves like the centralized license server in the conventional DRM system, even though the underlying licensing mechanisms are different. Details for using LAs for

issuing a license in the proposed DRM system will be given in subsequent subsections.

If a network supports PKI, PLI can be easily built on top of PKI. Each CA in PKI is assigned with additional task to store and maintain partial secrets and to provide public license services. In other words, each CA is extended in functions to be a LA too.

PLI is distributed and intrusion-tolerant. Even some LAs in a PLI are compromised, as long as the number of compromised partial shares is less than *k*, the system is still secure. This intrusion tolerance in PLI is similar to the fault tolerance provided by a P2P network. PLI is in fact a distributed trust architecture.

In the following, we shall describe the detail of the PLI based DRM system for P2P networks we propose in this paper.

3.2 Content Packaging and Distribution

Content packaging in our DRM system is the same as in the conventional DRM system which is described in Section 2.1. In this stage, a strong symmetric encryption algorithm such as the Advanced Encryption Standard (AES) [26] is used to encrypt the content.

Because our DRM system is built for P2P networks, content distribution is efficient and trivial. A publisher delivers the packaged content into a P2P network and the content is replicated to many nodes. A consumer can use the inherited search mechanism in a P2P network to easily locate and retrieve the desired content. The replication and cache mechanisms in a P2P network provide a good fault tolerance.

3.3 Establish Pre-license and Secret Sharing

A license contains the decryption key to unlock the content and some access rules or rights expressed in certain languages. Many well-known languages can be used to express rights, for example XRML (eXtensible Rights Markup Language) [27], XACML (eXtensible Access Control Markup Language) [28], ODRL (Open Digital Rights Language) [29], etc.

There are two types of licenses in our DRM system. The first type of license is called the *pre-license*, which is generated at the content packaging stage and will be used to generate the other type of license. The second type of license is called the *formal-license*, which is the license a consumer uses to play the protected content. The pre-license contains the decryption key associated with the access rules that the publisher of the content allows. The formal-license contains the decryption key and the access rules that a consumer, the owner of the formal-license, has. The formal-license will be described in the next subsection.

The pre-license, denoted as *prel* in the following, is encrypted with an asymmetric encryption algorithm such as RSA [30] and a public key *PK:*

$$prel = (license)^{PK}. \tag{1}$$

The corresponding secret private key *SK* is divided into *m* shares using a (*k*, *m*) threshold secret sharing scheme, which will be described in detail next. The publisher chooses *m* LAs and uploads one partial secret share to one chosen LA, along with the pre-license *prel* and the license ID. It is also possible to upload more than one secret share to a LA, or to choose more than *m* LAs and upload one secret share to multiple LAs. Recall that we have assumed that there exist trusted nodes in a network and these nodes operate as LAs for the network. Addresses of the partial secret holding LAs and other information are packaged with the content to facilitate a consumer to locate right LAs in generating the formal-license.

To obtain the *m* partial secrets, the following steps are performed:

i. The publisher generates the sharing polynomial

$$f(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1} \tag{2}$$

over a finite field Z_N where $a_0 = SK$.

ii. Each LA, identified by $id_i, i = 1, \cdots, m$, of the *m* chosen LAs is securely uploaded with the polynomial share

$$S_i = f(id_i) \mod N. \tag{3}$$

iii. The publisher broadcasts the *k* public witnesses of the sharing polynomial's coefficients $\{g^{a_0}, \cdots, g^{a_{k-1}}\}$, where $g \in Z_N$ after which it destroys the polynomial and quits.

iv. Each LA id_i verifies validity of the received share by checking if the following equation holds:

$$g^{S_i} = g^{a_0} \cdot (g^{a_1})^{id_i} \cdot \ldots \cdot (g^{a_{k-1}})^{id_i^{k-1}} \tag{4}$$

3.4 Formal-License Acquisition and Playing the Content

When a consumer retrieves the protected content and tries to play, the player checks whether a valid formal-license for the content is available at the local machine, and also checks if the content can be accessed. If these checks are OK, the player plays the content. Otherwise the player finds the LAs from the packaged content and contacts *k* live LAs for generating a formal-license for the consumer. The consumer might be involved for information registration or for payment which LAs can facilitate. After receiving the request, each LA responds with the partial result generated from its partial secret share. After receiving *k* correct partial results, the player combines them and generates a formal-license which typically binds to the specific machine the player runs. The player checks the access rules in the formal-license and plays the content. Typically all the DRM related operations at a local machine are performed by a black-box DRM module inside or coupled with the player. This module should be secure and tamper-proof.

Fig. 1 shows an example using a (*2*, *3*) threshold secret sharing scheme with three LAs. In this case, a failure of one LA does not affect the proper function of the system.

The detail of generating the formal-license is described below:

i. The node *p* where the player runs gets the list of partial secret holding LAs and contacts *k* live LAs with the license ID and the formal-license acquisition request.

ii. Each LA id_i calculates the partial result $prel_i$ from its partial secret share S_i :

$$prel_i = (prel)^{S_i} \bmod N . \tag{5}$$

It generates a random number u and calculates $A_1 = g^u$, $A_2 = prel^u$, $r = u - c \cdot S_i$, and

$$c = hash\,(g^{S_i}, prel_i, A_1, A_2) . \tag{6}$$

The LA responds by sending $prel_i$, A_1, A_2, and r securely to the requesting node *p*.

iii. The node *p* calculates

$$g^{S_i} = g^{a_0} \cdot (g^{a_1})^{id_i} \cdot \ldots \cdot (g^{a_{k-1}})^{id_i^{k-1}} \tag{7}$$

from the public witnesses of the sharing polynomial's coefficients. Eq. (6) is then applied to g^{S_i} and the received $prel_i$, A_1, A_2 to calculate *c*. The received partial result $prel_i$ is verified by checking if the following equations hold: $g^r \cdot (g^{S_i})^c = A_1$ and $prel^r \cdot (prel_i)^c = A_2$. The above steps are repeated until the node *p* gets *k* valid partial results. If less than *k* valid partial results can be obtained, the formal-license generation fails.

iv. The node *p* uses the *k* valid partial results to calculate the license:

$$\begin{aligned} license &= \prod_i (prel_i)^{l_{id_i}(0)} = (prel)^{\sum_i S_i \cdot l_{id_i}(0)} \\ &= (prel)^{SK} = ((license)^{PK})^{SK}, \end{aligned} \tag{8}$$

where $l_{id_i}(x) = \prod_{j=1, j\neq i}^{k} \frac{x - id_j}{id_i - id_j}$. The access rules in the license may be modified to reflect the actual rights the consumer gets. The license is then encrypted with some secret key related to the specific hardware of the node *p* to generate a formal-license, an individualized license that can be used only by the machine. The formal-license is stored in the local machine for future access.

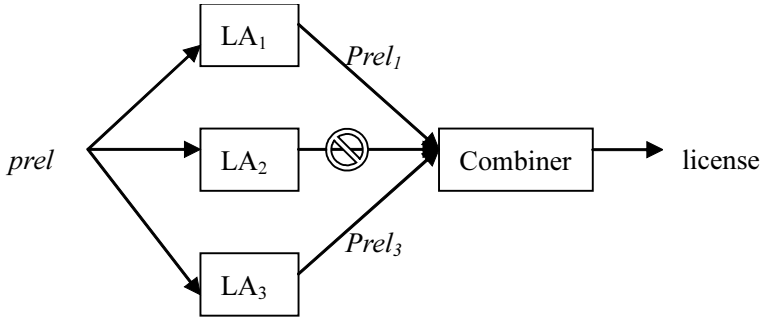

Fig. 1. An example with a (*2, 3*) threshold scheme where failure of one LA does not affect the normal operation.

3.5 Proactive Shares Update

In the secret sharing scheme described above the secret is protected by distributing partial secrets among LAs. Given sufficiently long time an attacker may finally compromise *k* partial secrets to deduce the secret key *SK*. To thwart such an attack, the partial secret shares are updated periodically with a proactive secret sharing scheme. An attacker Bob has to compromise *k* partial secrets before the partial secrets are updated. Otherwise he has to restart his attacks again. The proactive secret share update algorithms proposed in [18][19][20] can be applied to create a community of *m* entities with the new version of secret shares.

At periodic intervals the LAs update their shares of the private key *SK*. At the beginning of an update, each LA *i* generates a random *(k, m)* sharing of the secret *0* using a random update polynomial $f_{i,update}(x)$:

$$f_{i,update}(x) = b_{i,1}x + ... + b_{i,k-1}x^{k-1} \bmod N \tag{9}$$

Then LA *i* calculates the subshares $S_{i,j} = f_{i,update}(j)$, $j = 1,\cdots,m$, and distributes $S_{i,j}$ to all LA *j*, $j = 1,\cdots,m$.

Now each LA *i* has *m* subshares $S_{j,i}$, $j = 1,\cdots,m$ from all LAs. These subshares are added to the original share S_i, and the result is the new updated share:

$$S_i' = S_i + \sum_{j=1}^{m} S_{j,i} \,. \tag{10}$$

The corresponding new secret sharing polynomial $f_{new}(x)$ is the summation of the original polynomial $f(x)$ and all the randomly generated polynomials $f_{i,update}(x)$. It can be seen from the following proof that S_i' is indeed the partial secret share generated from $f_{new}(x)$.

Proof:

$$f_{new}(x) \equiv f(x) + \sum_{j=1}^{m} f_{j,update}(x) = a_0 + (a_1 + \sum_{j=1}^{m} b_{j,1})x + ... + (a_{k-1} + \sum_{j=1}^{m} b_{j,k-1})x^{k-1} \bmod N,$$

$$S_i' = S_i + \sum_{j=1}^{m} S_{i,j} = f(i) + \sum_{j=1}^{m} f_{j,update}(i) = f_{new}(i).$$

Figure 2 illustrates the share update scheme.

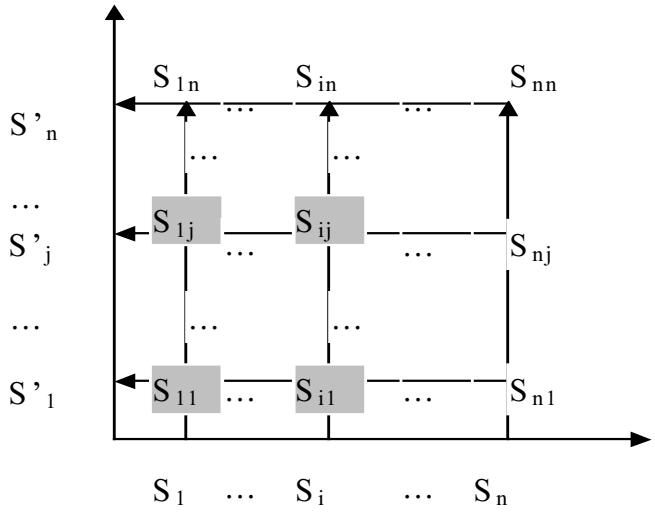

Fig. 2. The proactive share update scheme.

4 Discussion

4.1 Reliability and Intrusion Tolerance

Reliability and intrusion tolerance of the proposed system rely on the three factors: strong cryptographic algorithms, replication and caching mechanisms of the P2P network, and a (*k*, *m*) threshold secret sharing scheme.

Strong encryption algorithms such as AES and RSA provide robust content protection and ensure that only the authorized users can access the content.

Many P2P networks provide desirable properties such as replication and caching. Copies of the content are well distributed in the system. If a node is unavailable or a file is unavailable in some nodes, there are still many other nodes or file copies in the system which guarantees fault tolerance of the system.

Using a (*k*, *m*) threshold secret sharing scheme, the secret is divided into many partial secrets and distributed to many LAs. Attackers have to compromise *k* or more LAs to break the system. When *k* is large enough, the proposed system is really intrusion-tolerant.

4.2 Security of the DRM Proposed System

There are many communication sessions between nodes and LAs and among LAs. It is necessary to protect the security of these communication channels. The Secure Sockets Layer (SSL) [31] is used to ensure the communication security. LAs carry certificates to protect them from being impersonated by attackers. The partial result verification mechanism in the step 3 of formal-license generation described in Section 3.4 also thwarts invalid responses. The DRM module running at the consumer's machine also plays a critical role for the security of the system. Its security is exactly the same as the conventional DRM system so the technologies used in the conventional DRM system can also be used in our system.

5 Conclusion

In this paper, we have proposed the Public License Infrastructure (PLI) and the License Authorities (LAs) which are used to build a distributed DRM license service system. Based on PLI and LAs, a novel distributed DRM system has then been proposed for Peer-to-Peer networks. The proposed system uses a (*k*, *m*) threshold secret sharing and proactive shares update. The threshold secret sharing and the distributed PLI make the proposed system intrusion-tolerant, fault-tolerant, flexible, scalable, reliable, and highly available. Complex and centralized license servers in a conventional DRM system are no longer needed. The license service in the proposed system is provided collectively by group of redundant LAs.

References

1. R. Iannella, "Digital Rights Management (DRM) Architectures," *D-Lib Magazine*, vol. 7, no. 6, June 2001.
2. A. M. Eskicioglu, J. Town, and E. J. Delp, "Security of Digital Entertainment Content from Creation to Consumption," *Signal Processing: Image Communication, Special Issue on Image Security*, vol. 18, no. 4, April 2003, pp. 237–262.
3. Adobe EBooks, available at http://www.adobe.com/epaper/ebooks.
4. Intertrust, available at http://www.intertrust.com.
5. Microsoft Windows Media Digital Rights Management, available at http://www.microsoft.com/windows/windowsmedia/drm.aspx.
6. Napster website, available at http://www.napster.com.
7. Gnutella website, available at http://www.gnutella.com.
8. Architecture of Windows Media Rights Manager, available at http://www.microsoft.com/windows/windowsmedia/wm7/drm/architecture.aspx.
9. Napster, available at http://www.napster.com.

10. Gnutella website, http://www.gnutella.com.
11. JXTA, available at http://www.jxta.org.
12. I. Clarke, B. Wiley, O. Sanberg, and T. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Proc. Int. Workshop on Design Issues in Anonymity and Unobservability*, Springer Verlag, LNCS 2009, 2001.
13. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord A Scalable Peer-to-peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM'01*, San Diego, California, USA, 2001.
14. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM'01*, San Diego, California, USA, 2001
15. A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," *IFIP/ACM Int. Conf. Distributed Systems Platforms (Middleware)*, 2001.
16. B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-Area Location and Routing," *Technical Report No. UCB/CSD-01-1141*, Univ. of California, Berkeley.
17. A. Shamir, "How to Share a Secret," *Communications of ACM*, vol. 24, no. 11, 1979, pp. 612–613.
18. V. Shoup, "Practical Threshold Signatures," *Proc. EUROCRPT'00*, Springer Verlag, LNCS 1807, 2000, pp. 207–220.
19. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing, or: How to Cope with Perpetual Leakage," *Proc. CRYPTO'95*, Springer Verlag, LNCS 963, 1995, pp. 339–352.
20. C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems," *Proc. 9th ACM Computer & Comm. Security*, Washington D.C., 2002.
21. T. Wu, M. Malkin, and D. Boneh, "Building Intrusion Tolerant Applications," *Proc. 8th USENIX Security Symp*., pp. 79–91, 1999.
22. L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks", *IEEE Networks*, vol. 13, no. 6, 1999, pp. 24–30.
23. H. Luo and S. Lu, "Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks", *Technical Report 200030*, Dept. of Computer Science, Univ. Of California, Los Angeles, 2000.
24. J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks," *IEEE 9th Int. Conf. Network Protocols*, Nov. 2001, pp. 11–14.
25. H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-Securing Ad Hoc Wireless Networks", *IEEE 7th Int. Symp. Computers & Comm*. July 2002, pp. 567–574.
26. The Advanced Encryption Standard (AES), *FIPS-197*, also available from http://csrc.nist.gov/CryptoToolkit/aes/
27. eXtensible Rights Markup Language (XrML), available at http://www.xrml.org/
28. Extensible Access Control Markup Language (XACML), available at http://xml.coverpages.org/xacml.html.
29. Open Digital Rights Language (ODRL), available at: http://www.odrl.net.
30. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed., John Wiley & Sons, Inc. 1996.
31. SSL 3.0 Specification, available at http://wp.netscape.com/eng/ssl3/.

Improved Algebraic Traitor Tracing Scheme

Chunyan Bai and Guiliang Feng

Center for Advanced Computer Studies,
University of Louisiana at Lafayette,
Lafayette, LA 70504, USA
{cxb7146, glf}@cacs.louisiana.edu
http://www.cacs.louisiana.edu

Abstract. In this paper, we try to use algebraic-geometric codes (AG codes) to solve the traitor tracing problem over the broadcast channel. The scheme is shown by using AG codes to construct the linear space tracing code Γ, which is the base for the distributor to create private keys for each authorized subscribers. The obtained public key tracing scheme is deterministic and can trace all the participated traitors. Compared to the Reed-Solomon code (RS code) based public key traitor tracing scheme, our scheme can accommodate more users and tolerate more colluders given a fixed length of private keys.

Index Terms: algebraic-geometric code, Reed-Solomon code, traitor tracing, broadcast.

1 Introduction

With the rapid development of new IT technologies and electronic commerce, broadcast encryption is used in a wide range of situations such as video-on-demand, multi-party teleconferencing, stock quote distribution and updating softwares. In these cases, everyone can receive the encrypted message, but only a set of registered users are authorized to decrypt and discover the original message. This is implemented by using a public key to encrypt the message and broadcast it into the channel. Also, different decryption keys are generated and distributed to each legitimated user to guarantee him to obtain the subscribed service.

The authorized users, however, may have the temptation to distribute their description keys or the decrypted message further to unauthorized users without the permission of the distributor. This message leakage will cause great loss for the distributor. Such unauthorized access to data is known as *piracy*. Those authorized users who allow other non-authorized users to obtain data are called *traitors* and those unauthorized users are called *pirate users*. This problem can be solved by assigning different decryption keys to different user, then the compromised key could be used to trace back to its origin. While, a new problem will come. If a group of users try to collude and create a new decryption key, which may hide the identity of each colluders. How to prevent such message leakage is the key for the security of broadcast communication.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 217–227, 2003.

There are two non-exclusive approaches for the distributor to protect himself from the non-authorized redistribution. One option is to deter users from revealing their personal keys to others, which is referred to as *self enforcement*; Another option is to trace the corrupt users back, which is known as *traitor tracing*, and revoke them from further using the service. The self enforcement property is obtained by inserting some sensitive private information, such as bank account or credit card number, of traitors into his personal keys. Then the traitor will feel reluctant to redistribute his personal key to others. Although self enforcement schemes can prevent small scale piracy and can make it harder for pirates to obtain decryption keys, we will concentrate on traitor tracing in this paper since we assume that traitors DO want to reveal their keys to unauthorized users.

Traitor tracing refers to the task of identifying the keys used for generating the pirate key. In most of the available traitor tracing schemes, the keys have some combinatorial properties and tracing is probabilistic[1-7,13,14,18]. These work follows the traitor tracing model proposed by Chor,Fiat and Naor[1]. Each message is encrypted by a public key and sent to the channel together with different private keys for each individual user. Tracing is based on the combinatorial properties of the keys. It is guaranteed from these schemes that at least one of the traitors will be traced with high probability if less than k traitors participate in the collusion, which is called k-collusion resistant. Further work have been done to make the scheme full tracing[10,15,17]. These schemes are deterministic and catches all of the traitors who contributed to the attack based on the number theoretic assumption. Dynamic traitor tracing schemes in [8,9] are designed to combat the less common scenario where a pirate publishes the periodical access control keys on the Internet or simply rebroadcasts the content via an independent pirate network. Authors in [12,13] discuss how to revoke those traitors after they have been discovered. Recent work in [16,18,19] try to break some of the available traitor tracing schemes and propose new algebraic schemes by using error-correcting encoding and decoding. We will review the related work in Section 2.

In this paper, algebraic-geometric codes are used to construct the base for the public key traitor tracing scheme. The obtained scheme accommodate more users and achieve a higher threshold of tracing compared to the existed Reed-Solomon code based scheme.

The rest of this paper is organized as follows: In section 2, we describe work related to traitor tracing. Section 3 outlines an overview of the problem as well as definitions used in our approach. Section 4 characterizes and discusses the new algebraic-geometric codes based traitor tracing scheme. Finally, we summarize the conclusion and future work in section 5.

2 Related Work

The first attempt to deal with the traitor tracing problem was proposed by Chor, Fiat and Naor[1] and generalized by Stinson and Wei in [5]. According to Chor et.al., k-resilient traceability schemes can be implemented, that is, at least one

traitor will be revealed on the confiscation of a pirate decoder if there are at most k traitors. Although Chor's work gives the model for solving the traitor tracing problem, their scheme is inefficient and non-constructive. Stinson and Wei showed some explicit constructions by using combinatorial designs, their work is better for small values of k and n, where k is the number of traitors and n is the number of users. Both of these two schemes are private key encryption scheme, that is, they are symmetric in the sense that legitimate users of the broadcast information share all their secrets with the information provider. Thus they cannot provide non-reputation. Pfitzmann [2] pointed out this problem and introduced asymmetric traceability schemes in which the information provider cannot frame an innocent user and no user can abuse the system without being detected. In [6,11,14,18], asymmetric traitor tracing schemes are further discussed and designed which make traitor tracing designs more practical in the case of disputation between information provider and users.

Traitor tracing scheme can be designed to operate against any pirate decoder with $non - negligible$ success probability, which is called *Fully resilient schemes*. Although the security quality of the fully resilient schemes are good (they perform better than breaking the underlying encryption system), their complexity costs are too high under some circumstances. Naor and Pinkas introduced threshold tracing schemes in [3] which can trace the source of keys of pirate decoders with probability greater than some threshold. These schemes present a dramatic reduction in the overhead compared to fully resilient schemes while provide quality that are good enough for most cases.

After noticing that all the discussed traceability schemes assume that the data supplier should assign the keys after he has determined whom the authorized users are, which maybe unreasonable because changes between authorized and unauthorized users might be frequent, Stinson and Wei [4] investigated the key preassigned traceability schemes in which the personal keys can be assigned before the authorized users are determined. The schemes have better traceability and are more efficient in the sense of information rate and broadcast information rate. [8] and [9] tried to combat the less common scenario where a pirate publishes the periodically access control keys on the Internet or simply rebroadcasts the content via an independent pirate network. The schemes are accomplished by using the watermarking techniques, which allows the broadcaster to generate different versions of the original content, with no noticeable degradation in the content quality. Watermarks found in the pirate copy are used to trace its supporting traitors. These schemes can deal with not only the case that the private keys are leaked, but also the case that the private messages are rebroadcasted.

In all the above work, traitor tracings are designed in symmetric encryption system, thus requires the information provider be coincident with the administrator of the secure broadcasting infrastructure. However, it is highly desirable to divide these roles, thus the appearance of public-key traitor tracing schemes in [6,7,10,11,18]. Public-key traitor tracing schemes can overcome the problem of non-reputation. The first public-key traceability scheme was shown in [6]. But

this scheme was broken by Stinson and Wei in [4], and Boneh and Franklin in [10].

The next public key tracing scheme came with Boneh and Franklin in 1999[10]. The construction of keys is algebraic, which combines the theory of error correcting codes to discrete logarithm representation problems, and tracing is deterministic. The scheme gains full tracing, that is, if at most k traitors have participated in generating a new key, they can all be traced. Moreover, the tracing algorithm is error free. Innocent users are never blamed. Furthermore, the scheme is efficient in complexity. Following the same idea, some other work [15,17] have been done. [15] presented a general attempt to make the scheme long-lived, that is, the server can adapt its encryption procedure to the presence of pirate decryption keys and modify the transmission to reach the legitimate users only. [17] gave an attack on [10] and proposed efficient modified scheme to make it robust. Our scheme in this paper is also based on the work in [10] because it is the only scheme which can trace all the traitors who participates in the message leakage.

Most recent work on traitor tracing are [16], [18] and [19]. The scheme in [16] is specifically designed for tracing fingerprint media data such as images and video data. Generalized Reed-Solomon codes and their soft-decision decoding algorithms are utilized to present a powerful tracing scheme. [18] broke two of previous tracing schemes and present a new asymmetric public-key traitor tracing procedure with detailed proof of the traceability and security. In [19], a coding theoretic approach is used to produce a fast traitor tracing scheme. It is shown that when suitable error-correcting codes are used to construct traceability schemes, and fast list decoding algorithms are used to trace, the runtime of the tracing algorithm is polynomial in the codeword length. Also, the question of what the information provider should do after finding the traitors is considered. [12] and [13] talk about how to combine tracing scheme with revocation scheme and make the broadcast information distribution more robust. This direction is of great significance for future research.

3 Overview

Traitor tracing schemes help in three aspects of piracy prevention: they deter users from cooperating with pirates, they identify the pirates and enable to take legal actions against them, and they can be used to disable active pirate users. We will address the (k, n) traitor tracing problem in this paper, that is, to identify the source of at least one traitor if there are at most k traitors among n authorized users. a (k, n)-traceability scheme has four components: key generation, an encryption algorithm, a decryption algorithm and a tracing algorithm.

The (k, n) public key traitor tracing scheme proposed by Boneh and Franklin [10] provided a deterministic FULL tracing approach, that is, it can catch ALL traitors without accusing any innocent users as long as the number of traitors is at or below a collusion bound k. The construction of the scheme combines the

theory of error correcting codes to discrete logarithm representation problems. Each private key is a different solution vector for a discrete logarithm problem with respect to a fixed base of field elements. The fixed base is used to construct the public encryption key. This was the first time that the idea of error-correcting code was combined with traitor tracing. The resulting scheme is more efficient than other available schemes in terms of algorithm complexity(the length of the keys) and the tracing performance. Next, we will summarize the idea of the so-called Boneh-Franklin scheme (BF scheme) discussed in [10].

First, let's review some definitions and assumptions that are useful for BF scheme and our own scheme.

Definition 1. (Representation[10]) *If* $y = \prod_{i=1}^{2k} h_i^{\delta_i}$, *we say that* $\boldsymbol{\delta} = (\delta_1, \delta_2, ..., \delta_{2k})$ *is a representation of* y *with respect to the base* $h_1, h_2, ..., h_{2k}$.

Definition 2. (Convex Combination) *We say that a vector* $\boldsymbol{d}$ *is the convex combination of vectors* $\boldsymbol{d_1}, \boldsymbol{d_2}, ..., \boldsymbol{d_m}$ *if* $\boldsymbol{d} = \sum_{i=1}^{m} \alpha_i \boldsymbol{d_i}$, *where* $\alpha_1, ..., \alpha_m$ *are scalars such that* $\sum_{i=1}^{m} \alpha_i = 1$.

According to [10], if $\boldsymbol{d_1}, \boldsymbol{d_2}, ..., \boldsymbol{d_m}$ are representations of y with respect to the same base and $\boldsymbol{d}$ is the convex combination of $\boldsymbol{d_1}, \boldsymbol{d_2}, ..., \boldsymbol{d_m}$, then $\boldsymbol{d}$ is also a representation of y.

The BF scheme works in a multiplicative cyclic group G_q, where q is a prime. The security of the scheme is based on the difficulty of the Decisional Diffie-Hellman problem(DDH) over G_q.

Definition 3. (DDH[20]) *Let* $g \in G_q$ *be a generator. Consider triples of the form R:* $\langle g^a, g^b, g^c \rangle$ *and triples of the form D:*$\langle g^a, g^b, g^{ab} \rangle$, *where* $a, b, c < order(g)$. *A predicate solves the DDH problem if it can distinguish the collection D from the collection R.*

Loosely speaking, the DDH assumption states that no efficient algorithm (polynomial time algorithm) can distinguish between the two distributions $< g^a, g^b, g^{ab} >$ and $< g^a, g^b, g^c >$. The DDH assumption is useful for constructing efficient cryptographic primitives with very strong security guarantees. These include the Diffie-Hellman key agreement protocol, the El Gamal encryption scheme and so on.

Next, let's see the four components in the construction of BF (k, n)-traceability scheme.

Let G be a $(n - 2k) \times n$ matrix as:

$$G = \begin{pmatrix} 1 & 1 & 1 \ldots & 1 \\ 1 & 2 & 3 \ldots & n \\ 1^2 & 2^2 & 3^2 \ldots & n^2 \\ . & . & . \ . & . \\ . & . & . \ . & . \\ 1^{n-2k-1} & 2^{n-2k-1} & 3^{n-2k-1} \ldots & n^{n-2k-1} \end{pmatrix} (mod\ q)\,. \tag{1}$$

Since any vector in the span of the rows of A corresponds to a polynomial of degree at most $n-2k-1$ evaluated at the points $1, 2, ..., n$, G is actually the generator matrix of a $(n, 2k)$ Reed-Solomon Code. Let $w_1, ..., w_{2k}$ be a basis of the linear space of vectors satisfying $G\boldsymbol{x} = 0 \ mod \ q$ and use these $2k$ vectors as the columns of a matrix, an $n \times 2k$ matrix Γ is obtained as

$$\Gamma = \left(\boldsymbol{w_1} \ \boldsymbol{w_2} \ \boldsymbol{w_3} \ ... \ \boldsymbol{w_{2k}} \right) \tag{2}$$

View the line vectors of matrix Γ as a set of codewords, the matrix Γ provides n codewords of length $2k$. Based on this set of codeword, the (n, k)-traceability schemes is designed as:

Key Generation : For $i = 1, ..., 2k$, the data supplier chooses a random $a_i \in Z_q$ and computes $y_i = g^{a_i}$, where g is a generator of G_q and $\sum_{j=1}^{2k} a_j \gamma_j^{(i)} \neq 0$. Then the public key is published as $< z, y_1, ..., y_{2k} >$, where $z = \prod_{i=1}^{2k} y_i^{\beta_i}$ for random chosen $\beta_1, ..., \beta_{2k} \in Z_q$. The personal decryption key of user i is computed as

$$\theta_i = (\sum_{j=1}^{2k} a_i \beta_j) / (\sum_{j=1}^{2k} a_j \gamma_j^{(i)}) \ (mod \ q),$$

where $\gamma^{(i)} = (\gamma_1^{(i)}, ..., \gamma_{2k}^{(i)}) \in \Gamma$ is the i'th codeword of Γ.

Encryption : For a message $M \in G_q$, the information provider computes the ciphertext as $h = (M * z^r, y_1^r, ..., y_{2k}^r)$, where $r \in Z_q$ is a random number.

Decryption : Each user i computes M from h as follows by using θ_i as:

$$M = M * z^r / U^{\theta_i}, \ where \ U = \prod_{j=1}^{2k} (y_j^r)^{\gamma_j^{(i)}}.$$

Traitor Tracing : It is indicated in [10] that if $\boldsymbol{d_1}, ..., \boldsymbol{d_m} \in Z_q^{2k}$ are representations of y, then the convex combinations are the only new representations of y that can be efficiently constructed from $\boldsymbol{d_1}, ..., \boldsymbol{d_m}$. Any representation $(\delta_1, ..., \delta_{2k})$ of y with respect to the base h_i can be used as a decryption key. This is because $\prod_{j=1}^{2k} (h_j^a)^{\delta_j} = y^a$. If the traitors form a new decryption key $\boldsymbol{d}$ by making the convex combination of at most k decryption keys $\boldsymbol{d_1}, ..., \boldsymbol{d_m}$, then those traitors who participated in forming $\boldsymbol{d}$ can be efficiently determined by finding a vector $\boldsymbol{w} \in F_q^n$ of Hamming weight at most k such that $\boldsymbol{w} * \Gamma = \boldsymbol{d}$. Berlekamp's algorithm is used in this procedure to find traitors.

The (k, n)-traceability scheme in [10] is based on the theory of Reed-Solomon codes and the problem of discrete log representation. Traceability follows from the hardness of discrete log. The private key in all cases is just a single element of a finite field and can be as short as 160 bits. The complexity of encryption and decryption is independent of the size of the coalition under the pirate's control. But the key will be long if the finite field is big.

Notice that for a Reed-Solomon code over the finite field $GF(q)$, the codeword length N has to be less than or equal to q, i.e., $N \leq q$. Conversely, for an algebraic-geometric code, the codeword length can be greater than q.

This presents the possibility of using algebraic-geometric codes instead of Reed-Solomon codes to further reduce the bits needed to represent the private keys and consequently the complexity of the scheme.

4 Traitor Tracing Based on Algebraic-Geometric Codes (AG Codes)

4.1 Background on Algebraic-Geometric Codes

We now give the definition of AG codes, also known as geometric Goppa codes following the notation in [21].

Let X be an absolutely irreducible smooth projective algebraic curve of genus g over the finite field $GF(q)$. Consider an ordered set $P = \{P_1, P_2, ..., P_n\}$ of distinct rational points on X and a divisor D on X, rational over $GF(q)$. For simplicity, let us assume that the support of D is disjoint from P.

Definition 4. (Algebraic – Geometric Code) *The linear space $L(D)$ of rational functions on X associated with D yields the linear evaluation map*

$$L(D) \to F_q^n$$

defined by

$$f \to (f(P_1), ..., f(P_n)).$$

The image of this map is a linear code $C_L(P, D)$, which we call an algebraic-geometric code, or a geometric Goppa code.

Guruswami and Sudan give an efficient decoding algorithm of RS code and AG code in [22]. For those who need more discussion about AG code, please refer to [23],[24] and [25].

4.2 AG Codes Based Traitor Tracing Scheme

In this section, we will discuss how to use Algebraic-geometric codes to construct the *linear space tracing code* Γ, which is used to create private keys for each user.

Hermitian curve based construction. An AG code can be constructed from affine plane curves [25]. Let $H \doteq \{h_1, h_2, ..., h_r, ..., h_v\}$ be a sequence of vectors in F_q^n, where $h_r \doteq (h_{r1}, h_{r2}, ..., h_{rn})$, and let $S(r)$ be the linear space over F_q spanned by the first r vectors of H. Let $\hat{H} \doteq \{\hat{h}_1, \hat{h}_2, ..., \hat{h}_\mu, ..., \hat{h}_u\}$ be a supplementary sequence and $S(r, u)$ be the linear space over F_q spanned by only the first r vectors of H and all the vectors of $\hat{H}$. In most cases, the supplementary sequence $\hat{H}$ may be empty, that is, $u = 0$. Let

$$H_r \doteq \begin{bmatrix} h_1 \\ h_2 \\ ... \\ h_r \end{bmatrix} = \begin{bmatrix} h_{11}\ h_{12}\ ...\ h_{1n} \\ h_{21}\ h_{22}\ ...\ h_{2n} \\ ...\quad ...\ ...\quad ... \\ h_{r1}\ h_{r2}\ ...\ h_{rn} \end{bmatrix}. \tag{3}$$

We define $H_r^* \doteq \begin{bmatrix} \hat{H} \\ H_r \end{bmatrix}$ to be a parity check matrix of a linear code over F_q, denoted by C_r. For a special case of $u = 0$, H_r^* reduces to H_r. Next we will show how to include the current AG codes to the above H sequence construction.

Let χ be an algebraic geometric curve with the genus g and let $P_1, P_2, ..., P_n$ and P_∞ be the set of rational points over a finite field F_q. Let $H \doteq \{f_1, f_2, ..., f_n\}$ be a sequence of functions and $\hat{H} \doteq \phi$, then the linear code C_r defined by H_r^* is an $(n, n-r)$ AG code $C_\Omega(D, G)$, where $D = P_1+P_2+...+P_n$ and $G = (r+g-1)P_\infty$.

Improved geometric Goppa codes can be constructed from algebraic-geometric curves based on a well-behaving sequence H [23]. The key of the construction lies in the definitions of *weight* $w(x)$ and *total order*. Next, we will show how to construct an improved geometric code from the Hermitian curves.

Let a location set LS be a set of all the rational points of the Hermitian curve $x^{q+1} + y^q + y = 0$ over $GF(q^2)$, there should have a total of q^3 elements in LS. Let $w(x) = q$ and $w(y) = q + 1$, the total ordering of monomials $x^{i_1} y^{i_2}$ can be determined. After deleting all the monomials linearly dependent on their previous monomials in H, a well-behaving sequence $H(|H| = q^3)$ will be obtained, from which a $(n, n-r, d^*) = (q^3, q^3-r, d^*)$ AG code can be constructed. H_r^* will be treated as the parity check matrix for $(n, n-r, d^*)$ AG code, where r is determined by the given designed minimum distance d^* according to [23]. And this $(n-r) \times n$ matrix will be utilized as the G matrix, which is further used to create the linear space tracing code Γ in the traitor tracing scheme. Let us see an example about the detailed construction of the H sequence.

Consider the Hermitian curve $x^5 + y^4 + y = 0$ over $GF(4^2)$. Let $w(x) = 4$ and $w(y) = 5$, then we have

$$H' = \{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, x^4, x^3y,$$
$$x^2y^2, xy^3, y^4, x^5, x^4y, x^3y^2, x^2y^3, xy^4, x^6,$$
$$y^5, x^5y, x^4y^2, x^3y^3, x^2y^4, x^7, xy^5, x^6y, y^6, x^5y^2, ...\}$$

After deleting all the monomials denoted by $y^{i_2}x^{i_1}$, which are linearly dependent on their previous monomials, a well-behaving sequence H is found to be:

$$H = \{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, x^4, x^3y, x^2y^2,$$
$$xy^3, y^4, x^4y, x^3y^2, x^2y^3, xy^4, y^5, x^4y^2, x^3y^3,$$
$$x^2y^4, xy^5, y^6, x^4y^3, x^3y^4, x^2y^5, xy^6, y^7, x^4y^4,$$
$$x^3y^5, x^2y^6, xy^7, y^8, x^4y^5, x^3y^6, x^2y^7, xy^8,$$
$$y^9, x^4y^6, x^3y^7, x^2y^8, xy^9, y^{10}, x^4y^7, x^3y^8, x^2y^9,$$
$$xy^{10}, y^{11}, x^4y^8, x^3y^9, x^2y^{10}, xy^{11}, y^{12}, x^4y^9,$$
$$x^3y^{10}, x^2y^{11}, y^{13}, x^4y^{10}, x^3y^{11}, y^{14}, x^4y^{11}, y^{15}\}.$$

$|H| = 64 = 4^3$. The weight sequence of H is:

$$W = \{0, 4, 5, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20,$$

$$21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,$$

$$33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,$$

$$45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,$$

$$57, 58, 59, 60, 61, 62, 63, 65, 66, 67, 70, 71, 75\}.$$

We can see that the weight sequence is not a continuous integer sequence, i.e., some gaps exist. For example, $1, 2, 3$ between 0 and 4, $6, 7$ between 5 and 8. Such gaps provide the Hermitian curve with a genus, which is related to the performance of the constructed AG code. That is, the smaller the genus is, the greater the efficiency of the linear codes C_r.

Suppose we want to construct an improved geometric Goppa code with length $n = 64$ and designed minimum distance $d^* = 6$ over $GF(4^2)$. Using the construction 2.1 from [23], we have $r = 7$ and thus $H_7^* = (1, x, y, x^2, xy, y^2, x^3, y^3, x^4)^T$. This $(n-r) \times n = 9 \times 64$ matrix is the parity check matrix for $(64, 55, 6)$ AG code and will be utilized as the G matrix in the public key traitor tracing scheme.

The tracing algorithm, that is, how to efficiently determine the unique set of vectors in Γ used to construct the pirate key $\boldsymbol{d}$ can be implemented by the decoding algorithm of AG code in [24].

5 Discussion and Comparison

Notice that for a Reed-Solomon code over the finite field $GF(q)$, the codeword length n has to be less than or equal to q, i.e., $n \leq q$. Conversely, for an algebraic-geometric code, the codeword length can be designed with any desired length n, which can be greater than q. Because the codeword length determines the total number of users in the system, algebraic-geometric code based scheme will accommodate more users than Reed-Solomon code based scheme if both the schemes are defined on the same finite field $GF(q)$.

Now, let us see the complexity of the given pubic key traitor tracing scheme.

Suppose n is the total number of users in the system and k is the bound of the number of traitors, that is, the maximum number of traitors in a confiscation. In the presented scheme, the size of the ciphertext is $2k + 1$ elements of the given finite field $GF(q)$ and the size of the user-key is $O(1)$, that is, only one finite field element. The encryption key size is $2k + 1$ elements of the finite field. So the number of bits used to represent the element in a finite field is one of the key factors that will greatly affect the overload of the scheme.

If we use the parity check matrix of $(n, 2k, d)$ Reed-Solomon code over $GF(q^3)$ as the G matrix in the (n, k)-traceability scheme, then the codeword length n is less than or equal to the order of the finite field, q^3. That is, the maximum number of users that the scheme can accommodate is $n = q^3$ (this comes from the construction of the linear space tracing code Γ). In order to accommodate the

same number of users in the system, we can instead use an algebraic-geometric code which is defined on the Hermitian curve

$$x^{q+1} + y^q + y = 0 \ \ over \ \ GF(q^2).$$

Since the given Hermitian curve has a total of q^3 roots pairs $(x_i, y_i), i = 1, 2, ..., q^3$, the codeword length of the corresponding algebraic-geometric code can be designed as q^3 although the code is designed on finite field $GF(q^2)$. That is, the algebraic-geometric code based tracing scheme can also accommodate q^3 users. As we know that in order to represent each element of finite field $GF(q^3)$, a total number of $3log(q)$ bits is needed. On the contrary, $2log(q)$ bits are enough to represent each element of finite field $GF(q^2)$. That is, if we use AG code defined on Hermitian curve $x^{q+1} + y^q + y = 0$ over $GF(q^2)$ to construct the linear space tracing code Γ, $log(q) \times (4k + 2)$ bits will saved compared to RS code based tracing scheme in order to create the cipertext and the encryption key. Also, $n * log(q) = q^3 \times log(q)$ number of bits will be saved for a total of q^3 users in the system. Consequently, the AG code based traitor tracing scheme greatly eliminates the complexity overload compared to the RS code based scheme.

6 Conclusion and Future Work

In this paper, we use algebraic-geometric code to construct the linear space tracing code Γ, which is the key step in public key traitor tracing scheme. The obtained public key tracing scheme is deterministic and can trace all the particpated traitors. Compared to the Reed-Solomon code based public key traitor tracing scheme, our scheme can accommodate more users and tolerate more colluders given a fixed length of private keys. Also, the complexity overload of the AG code based scheme has been greatly eliminated compared to the RS code based scheme, which makes the AG code based scheme more feasible in practice.

Although the scheme presented in this paper is an pubic key tracing scheme, it is not an asymmetric scheme. That is, non-reputation is not provided in this scheme. Also, what the information provider should do after figuring out the traitors is not discussed in this paper. So how to make the scheme more robust in the case of the disputation between the information provider and users and how to combine the given traitor tracing scheme with revocation are our future work.

References

1. B.Chor, A.Fiat and M.Naor, "Tracing Traitors", Proceedings of Crypto'94, LNCS 839, Springer-Verlag, Berlin 1994.
2. B.Pfitzmann, "Trials of Traced Traitors", Workshop on Information Hiding, Cambridge, UK, LNCS 1174, Springer-Verlag, 1996.
3. M.Naor and B.Pinkas, "Threshold Traitor Tracing", Proceedings of Crypto'98, LNCS 1462, Springer-Verlag, Berlin 1998.

4. D.R.Stinson and R.Wei, "Key Preassigned Traceability for Broadcast encryption," Selected Area in Cryptology, SAC'98, LNCS 1556, 1999.
5. D.R.Stinson and R.Wei, "Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes", J. on Discrete Mathematics, Vol.11, No.1, 1998.
6. K.Kurusawa and Y.Desmedt, "Optimum Trator Tracing and Asymmetric Schemes", LNCS 1403,1998.
7. K.Kurusawa ,M.Burmester and Y.Desmedt, "A Proven Secure Tracing Algorithm for the Optimun KD Traitor Tracing Scheme", Proceedings of Eurocrypt'98, 1998.
8. A.Fiat and T.Tassa, "Dynamic Traitor Tracing," J. of Cryptology, Vol.4, 2001.
9. O.Berkaman, M.Parnas and J.Sgall, "Efficient Dynamic Traitor Tracing", J. on Computing, vol.30, No.6, 2001.
10. D.Boneh and M.Franklin, "An efficient Public Key Traitor Tracing Scheme", Proc. of Crypto'99, 1999.
11. A.Kiayias and M.Yung, "Self Protecting Pirates and Black-Box Traitot Tracing", Proc. of the 21st Annual International Cryptology Conference, Crypto'01, LNCS 2139, 2001.
12. D.Naor, M.Naor and J.Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", Proc. of Advances in Cryptoloty, Santa Barbara, CA, Aug., 2001
13. M.Naor and B.Pinkas, "Efficient Trace and Revoke Schemes", Proc. of Financial Cryptography, Anguila, 2000.
14. E.Magkos, P.Kotzanikolaou and V.Chrissikopoulos, "An Asymmetric Traceability Scheme for Copyright Protection Without Trust Assumptions", LCNCS 2115, 2001.
15. S.Maki, "On Long-Lived Public-Key Traitor Tracing. First Steps", Proc.of the Helsinki University of Technology, Semenar on Network Security, Fall, 2000.
16. R.Safavi-Naini and Y.Wang, "Traitor Tracing for Shortened and Corrupted Fingerprints", ACM workshop on Digital Rights Management, 2002.
17. J.J.Yan and Y.Wu, "An Attack on A Traitor Tracing Scheme", Technical Report No.518, Computer Laboratory, Univ. of Cambridge, 2001.
18. A.Kiayias and M.Yung, "Breaking and Repairing Asymmetric Public-Key Traitor Tracing", ACM workshop on Digital Rights Management, DRM'2002, 2002.
19. A.Silverberg, J.Staddon and J.Walker, "Efficient Traitor Tracing Algorithms using List Decoding", Cryptology ePrint Archive: Report 2001/016, 2001.
20. D.Boneh, "The Decision Diffie-Hellman Problem", Proc. of the 3rd Algorithmic Number Theory Symposium, LNCS Vol.1423, Springer, 1998.
21. J.H.van Lint and G.van der Geer, Introduction to coding theory and Algebraic geometry, Birkhauser Verlag publisher, Boston, 1988.
22. V.Guruswami and M.Sudan, "Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes," IEEE Transactions on Information theory, vol.45, 1999.
23. G.L. Feng and T.R.N. Rao, "Improved Geometric Goppa Codes Part I: Basic Theory,", IEEE Transactions on Information Theory, vol.41, No.6, 1995.
24. G.L. Feng, V.K. Wei, T.R.N. Rao, and K.K. Tzeng, "Simplified Understanding and Efficient Decoding of a Class of Algebraic-Geometric Codes". IEEE Transaction on Information Theory, vol.40, No.4, Jul. 1994.
25. G.L.Feng and T.R.N.Rao "A simple Approach for Construction of Algebraic-Geometric Codes from Affine Plane Curves", IEEE Transaction on Information Theory, vol.40, No.4, Jul. 1994.

Common Vulnerability Markup Language

Haitao Tian, Liusheng Huang, Zhi Zhou, and Hui Zhang

Department of Computer Science, University of Science and Technology of China, Hefei, Anhui, China
tht@mail.ustc.edu.cn

Abstract. Discovering, disclosing, and patching vulnerabilities in computer systems play a key role in the security area, but now vulnerability information from different sources is usually ambiguous text-based description that can't be efficiently shared and used in automated process. After explaining a model of vulnerability life cycle, this paper presents an XML-based common vulnerability markup language (CVML) describing vulnerabilities in a more structural way. Besides regular information contained in most of current vulnerability databases, information about classification, evaluation, checking existence and attack generation is also given in CVML. So it supports automated vulnerability assessment and remedy. A prototype of automated vulnerability management architecture based on CVML has been implemented. More manageable vulnerability databases will be built; promulgating and sharing of vulnerability knowledge will be easier; comparison and fusion of vulnerability information from different sources will be more efficient; moreover automated scanning and patching of vulnerabilities will lead to self-managing systems.

1 Introduction

With the fast development of Electronic Commerce and Electronic Government worldwide, security of computer systems is attracting more and more attention. As the current state of the art in security is "penetrate and patch", vulnerabilities that can lead to violating security mechanisms of systems are the focus in the area of security. Attackers (or Hackers) seek for and take advantage of existing vulnerabilities on target system to subvert the confidentiality, integrity or availability of information resources while security administrators assess their systems periodically and patch known vulnerabilities timely. Obviously the earlier and the more we know about the vulnerabilities in our systems, the more secure will they be. Discovery of vulnerabilities in a complex software system requires experience, creativity and good knowledge of the system, and unfortunately most people do not have this expert knowledge. So it is important for people to disseminate and share vulnerability knowledge with one another.

Software vulnerabilities are discovered by security researchers and disclosed in varying fashions. These include public disclosures, security advisories and security bulletins from vendors. However the terms and formats currently used in the field of

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 228–240, 2003.

computer vulnerability tend to be unique to different individuals and organizations. Moreover much of this information from different sources is informal, text-based description and can not currently be efficiently combined and shared among people not to mention support automated in-depth process.

After explaining a model of vulnerability life cycle, this paper presents a common vulnerability markup language (CVML) that describes vulnerabilities in a more structural and precise way based on XML. CVML intends to provide a uniform way to express and exchange information in the whole process of vulnerability management. With CVML, more manageable vulnerability databases will be built, promulgating and sharing of vulnerability knowledge will be easier, combine and refinement of vulnerability information from different sources will be done more efficiently. Moreover automatic scan and patch of system vulnerabilities are well supported. This paper is organized as follows: in section 2 the background and related work are discussed, then high level view of CVML is proposed in section 3, we give the document type definition (DTD) of CVML in section 4 and potential applications with CVML in section 5, Finally conclusions and future work is presented.

2 Background and Related Work

Complex information and communication systems give rise to design, implementation and management errors. These errors can lead to vulnerabilities—flaws in an information technology product that could allow violations of security policy. The discovery and exploitation of system vulnerabilities is requisite in each successful intrusion. In other words no vulnerabilities, no penetration.

2.1 Vulnerability Life Cycle

In general, vulnerabilities appear to transition through distinct states as depicted in Fig.1: birth, discovery, disclosure, the release of a fix, publication, exploitation or patching. Denoting the flaw's creation, birth usually occurs unintentionally during a large project's development when designers or developers of systems make mistakes. When someone (system administrator, attackers or other researchers) discovers that a product has security or survivability implications, the flaw becomes a vulnerability. It is clear that, in many cases, original discovery is not an event that can be known, the discoverer may never disclose his finding. Disclosure reveals details of the problem to a wider audience. It may be posted to a vulnerability-specific mailing list such as Bugtraq[7], or other security organizations. Vulnerability is correctable when the vendor or developer releases a software modification or configuration change that corrects the underlying flaw. Vulnerability becomes public when a computer security incident response center like CERT [8] or corresponding vendor could issue a report or advisory concerning the vulnerability. Usually correction is prior to publicity in order to prevent abuse of vulnerabilities.

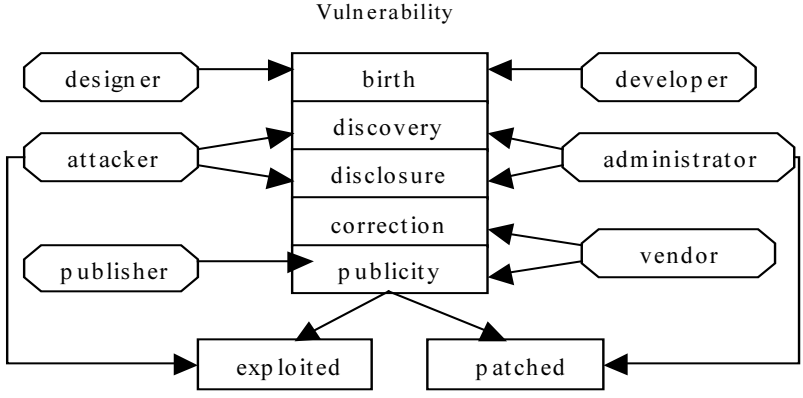

Fig. 1. Distinct states in vulnerability life cycle show a vulnerability-centered security architecture.

Attackers or security testers scan target systems, find vulnerabilities and exploit them for intrusions while security administrators take considerable time to get vulnerability information from security bulletins and scan their systems to patch known vulnerabilities timely. CERT or other computer security incident response teams respond, collect and publish vulnerability information. Software vendors are required to give advisories and patches of their products. In such a vulnerability-centered security architecture, lack of a consistent, structural and machine readable expression of vulnerability till now embarrasses automatic system scan and attack generation, interaction of different related products, combination and refining of vulnerability information from different sources. A specific example is six organizations have issued eight separate bulletins for one Microsoft Outlook vulnerability, with the time between the first and last bulletins spanning nearly a year. These bulletins variously described the vulnerability as the Outlook overrun vulnerability, MIME (Multipurpose Internet Mail Extensions) buffer overflow, MIME name vulnerability, long filename vulnerability, and mail tool vulnerability. The common vulnerability markup language we propose in next section can solve these problems.

2.2 Related Work

Most of the research on vulnerability has focused on discovery and collecting such as[7,8], there are also some meaningful studies on the classification of vulnerabilities in[2,9,10,11] which is the foundation of further description. To our knowledge, there is not a standard language to efficiently express vulnerabilities till now. But we can give some work toward the right direction.

The MITRE Corporation is well known for their creation of CVE[4]. The CVE is a dictionary of vulnerabilities that links vulnerability reports through a common naming system, but it doesn't include technical details or information on risk, impact, and

solutions. Still new and evolving, neither the lists of references nor the coverage of vulnerabilities is yet complete. It is more important to standardize all-sided descriptions of vulnerabilities than just names. Moreover the CVE names with the format CVE-YYYY-NNNN don't have a beneficial meaning. CVE also has a long screening process and a board to make final decisions, which makes it impossible to keep up with the flood of vulnerabilities.

MITRE just released a new standard for vulnerability assessment OVAL[5] in November, 2002. The open vulnerability assessment language is a common language for security experts to discuss and agree on technical details about how to check for the presence of a vulnerability on a computer system. The vulnerabilities are identified by OVAL queries like SQL that perform the checks. OVAL's purpose is only to check if software is vulnerable or not in other words in what conditions a vulnerability exists, which is only a part of our CVML. OVAL also faces the same problem of having a long screening process that might delay the release of vulnerability checks. However we can use OVAL in our CVML to check the existence of vulnerabilities.

The Open Web Application Security Project (OWASP) presented VulnXML in July, 2003 which is an open standard format for web application security vulnerabilities[6]. Anyone could describe web application security vulnerabilities with VulnXML so that the knowledge could be openly shared with both tools and users. VulnXML includes details about the vulnerability as well tests to check if the vulnerability exists. This is actually a step in the same direction as our CVML but VulnXML is for web application security vulnerabilities only.

The Intrusion Detection Exchange Format Working Group (IDWG) of the Internet Engineering Task Force is working on a common intrusion language specification [15] based on XML, which describes data formats for sharing information of interest to intrusion detection and response systems, and to management systems that may need to interact with them. However that work focuses on standardizing the information exported by intrusion detection systems (such as various alerts), but not on vulnerabilities themselves, and it is mainly to be used for automated intrusion detection.

3 Common Vulnerability Markup Language

CVML is an XML-based machine language to describe vulnerabilities in a uniform way for exchange and automatic process of vulnerability information. It is mainly composed of four sections of information about vulnerabilities: general related, judge-existence related, exploitation related, solution related.

3.1 Generally Related Information

In CVML general related information of vulnerabilities includes: ID, name, CVE name (optional), references, related dates, revision history, summary, classification, evaluation, additional note, credit. Most of these elements that are also included in

current vulnerability databases like [7,8] are easy to understand. However, we want to explain some characteristic elements.

As mentioned above the value of CVE name with the format CVE-YYYY-NNNN is little for organization and further process such as searching vulnerability for a specific product. In CVML the vulnerability name consists of a unique vendor identifier (e.g. MS for Microsoft, IBM for IBM, etc), the affected software name or the affected filename, the nature of the defect causing the problem and the impact. For instance "CVE-1999-0228" is named 'MS-NT-Service-Rpcss-InputValidation-DoS' in CVML. In this way we can organize the vulnerabilities into a forest-like namespace that facilitates classification and searching.

The element of classification demonstrates vulnerabilities in three aspects: how, where, when. How means the genesis of a vulnerability which can be one of these: Input Validation, Boundary Condition, Buffer Overflow, Access Validation, Exceptional Condition, Environmental Error, Configuration Error, Race Condition error, and other error etc. Where indicates the location of the vulnerability in the system (such as System Initialization, Memory Management, Process Management, Device Management, File Management, Identification and Authentication, Network Protocol, Support Software, Application, Hardware etc. when shows in what phase of the software life cycle the vulnerability is introduced (design, implementation, maintenance or operation). The taxonomy in CVML is based on previous work and contributes to the statistic and analysis of vulnerabilities that is important for the improvement of software design and development.

The element evaluation evaluates the vulnerability with three attributes: direct impact, total cost of exploit and rating. The direct impact tells the direct result caused by the vulnerability such as root access, user access, system down, disclosure of sensitive data etc. Total Cost of Exploit means the total resources (such as hardware and software resources) and skills (such as knowledge of specific system or the ability of programming with assembly on certain system) needed to exploit the vulnerability. It is approved that vulnerabilities with lower TOE incline to be more popular. Rating gives the severity of the vulnerability in the manner of quality or quantity. Generally speaking vulnerabilities with more severe direct impact and less total cost of exploit are more serious. Precise evaluation scheme of vulnerabilities is an interesting topic, but not the theme of this paper.

3.2 Check-Existence Related Information

The element check-existence provides information about how to check for the presence of vulnerability on a computer system. It includes three sub-elements: affected platform, affected software, and affected file to indicate where the vulnerability exists. Affected platform is composed of operating system and architecture (such as sparc, intel x86). Affected software consists of name and version. Affected file has two sub-elements: path and name. OVAL queries [5] can also be included for compatibility. Information here contributes to automatic vulnerability scan by security assess tools.

3.3 Solution Related Information

The section of solution contains information about how to correct the vulnerability. It consists of two parts: workaround and patch. Workaround provides operations taken in the absence of patching to avoid the vulnerability (such as disable vulnerable service, restrict buffer size with a system management tool, make specific configuration of the software etc). The element “configuration” in workaround can be classified into registry-configuration mainly referring to software running on Windows and file-registration referring to others. The element “registry configuration” contains registry key, name, type, data of a registry entry. The element “file configuration” demonstrates an entry in a designated file. The reference URL refers to sites for more information. Patch gives the patch name, a short note and the update URL pointing to a download site or a web service to fix the vulnerability, which can enable automatic update or hot-fix for system security. The sub-element patch-trace of patch specifies how to check whether the patch has been installed on the target system. It can include an entry name, an entry value of certain registry key for MS Windows, it also can refer to some specific entry in a specific file such as “inetd.conf” in Solaris. The sub-element patch-trace also contributes to system scan for vulnerabilities because a specific patch indicates the non-existence of one or more vulnerabilities. So we also can prevent updating software repeatedly in automatic system patching.

3.4 Exploit Related Information

The element of exploit is from the view of attackers or security testers and designed to support the automatic attack illation and generation. We can model attacks with 2-layer topology: functionality-level referring to concrete atomic vulnerability exploit and concept referring high-level logical description for illation. The exploit element contains three child elements: precondition, procedure, post-condition. Precondition and post-condition contribute to concept level description while procedure represents the functionality level.

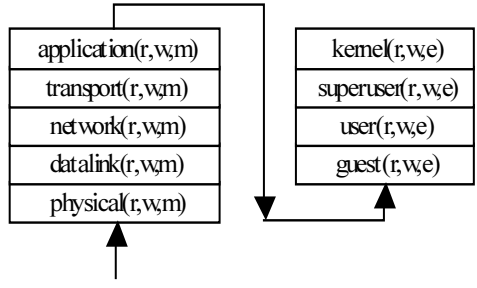

Fig. 2. Privilege Chaining

Precondition and post-condition gives both the privilege required or got on the target system for exploit of the vulnerability and high-level concepts in form of any attack description languages such as in [12,13]. As most of current operating systems are based on TCP/IP network, so we present a stratified structure of privileges shown in Fig.2. The right Modify on certain layer means the ability of arbitrary modification or redirection of entities on the layer. A successful computer intrusion consists of a few phases. An attacker increases his privilege on the target system in each phase of the attack until he gets the ultimate goal: super user privilege or absolute control of the system. For example an attacker will have the privilege Transport (Write) for a system without a firewall, which means he can write packets to the transport layer. He can use Nmap[14] to increase his privilege to Application (Read) that means he knows the services running on the target system. Then he can search for vulnerabilities in these applications running to elevate his privilege continually. Assuming vulnerability A can lead to local user access with remote access and vulnerability B can lead to super user access with local user access, therefore exploit sequence A, B would provide root access. In other words we can chain concrete vulnerabilities according to the privilege needed (in precondition) and got (in post-condition) to find a sequence of exploits taken for an appointed privilege, which we call privilege chaining. Privilege chaining can be used in attack generation although it is of a big granularity. We can also model a specific Operating System with finely granular privileges and make precise conclusion in intrusion through privilege chaining. Detail about definition and ratiocination of vulnerabilities in concept level is beyond the scope of this paper and it is confirmed that any study results in concept level can be used in CVML for ratiocination in automatic attack in a more formal and accurate way than privilege chaining.

Procedure gives the functionality-level steps taken by the attacker in the form of any attack description languages or source code in programming language such as c, c++, Perl etc. Compilation and execution of this part by the attacker will exploit the vulnerability and it can be called as a module by the attack engine in the concept-level based on the ratiocination made in the concept level.

4 DTD of CVML

We give the complete document declaration (DTD) of CVML and it is easy to understand with the comments.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT vulnerability (VId, VName, CVEName?, Summary, Date, History?, Reference*, Classification, Evaluation, CheckExistence, Solution?, Exploit, VNote, Credit)>

<!ELEMENT VId (#PCDATA)>
<!--This is a unique identifier number assigned to the vulnerability like “200305080001” -->
```

```
<!ELEMENT VName (#PCDATA)>
<!--This is the name in CVML like “MS-Windows-IE5.0-bufferoverflow”-->
<!ELEMENT CVEName (#PCDATA)>
<!--This is the CVE name if it exists-->
<!ELEMENT Summary (#PCDATA)>
<!--This is an abstract of the vulnerability -->
<!ELEMENT Date EMPTY>
<!ATTLIST date
discovery  CDATA #REQUIRED
firstPublish  CDATA #REQUIRED
lastModified  CDATA #REQUIRED>
<!--This field tells the related dates of discovery, publish and last modified -->
<!ELEMENT  History (Revision+)>
<!ELEMENT  Revision (RevisionDate, RevisionNote)>
<!ATTLIST  Revision version CDATA #REQUIRED>
<!ELEMENT RevisionDate (#PCDATA)>
<!ELEMENT RevisionNote (#PCDATA)>
<!--  This field is intended to record the revision history of the vulnerability -->
<!ELEMENT Reference (#PCDATA)>
<!ATTLIST Reference
            Source  CDATA #REQUIRED
            URL  CDATA #IMPLIED >
<! --This field provides resources such as Bugtraq bid’s, CERT announcements etc. URL should point directly to the entry of the particular vulnerability if possible -->

<!ELEMENT Classification (Genesis, Location, Introduce)>
<!ELEMENT Genesis (#PCDATA)>
<!--This indicates the genesis of the vulnerability, its value can currently be one of the following: Input Validation, Boundary Condition, Buffer Overflow, Access Validation, Exceptional Condition, Environmental Error, Configuration Error, Race Condition error, and other error etc.-->
<!ELEMENT Location (#PCDATA)>
<!--This indicates the location of the vulnerability in computer system. It can be one of the following: System Initialization, Memory Management, Process Management, Device Management, File Management, Identification and Authentication, Network Protocol, Support Software, Application, Hardware etc. -->
<!ELEMENT Introduction (#PCDATA)>
<!-- This field indicates when the vulnerability is introduced in the system. It can be one of the following: design, implementation, maintenance or operation -->

<!ELEMENT Evaluation EMPTY>
<!ATTLIST Evaluation
            Impact CDATA #REQUIRED
            AttackCost CDATA #REQUIRED
            Rating CDATA #REQUIRED>
```

<!-- This field evaluates the vulnerability in 3 aspects, Impact can be root access, user access, system down, disclosure of sensitive data etc. Attack cost or rating now can be high, medium, low. They also can be a number got through certain evaluation scheme. -->

<!ELEMENT CheckExistence (Platform+, AffectedSoftware+, Affectedfile*) >
<!ELEMENT Platform (OS,Arch)>
<!ELEMENT OS (OSName, OSVersion)>
<!ELEMENT OSName(#PCDATA)>
<!ELEMENT OSVersion(#PCDATA)>
<!-- This should be the official prodocut name and version such as Microsoft Windows NT Workstation 3.51 -->
<!ELEMENT Arch (#PCDATA)>
<!-- This should be the official architecture string such as i386, ia64 , ppc etc. If the vunerability exists in all the architecture running under certain OS, it can be blank. -->
<!ELEMENT AffectedSoftware (SoftwareName, SoftwareVersion)
<!ELEMENT SoftwareName (#PCDATA)>
<!ELEMENT SoftwareVersion (#PCDATA)>
<!--This gives name and version of the affected software -->
<!ELEMENT AffectedFile (FileName, Filepath, fileattribute?)>
<! ELEMENT FileName (#PCDATA)>
<!ELEMENT FilePath (#PCDATA)>
<!ELEMENT FileAttribute (#PCDATA)>
<!-- This section gives the affected file if necessary-->

<!ELEMENT Solution (Workaround*, Patch*)>

<!ELEMENT WorkAround (WorkNote, Configuration?, WorkUrl*)>
<!ELEMENT WorkNote (#PCDATA)>
<!-- This is the note of the workaround.-->
<!ELEMENT Configuation (RegistryCfg*,FileCfg*)>
<!ELEMENT RegistryCfg (CfgKey, CfgName, CfgType, CfgData)>
<!ELEMENT CfgKey(#PCDATA)>
<!ELEMENT CfgName(#PCDATA)>
<!ELEMENT CfgType(#PCDATA)>
<!ELEMENT CfgData(#PCDATA)>
<!-- This indicates the registry entry for software configuration on Windows in the workaround to wipe off the vulnerability-->
<!ELEMENT FileCfg (CfgPath, CfgName, CfgEntry)>
<!ELEMENT CfgPath(#PCDATA)>
<!ELEMENT CfgName(#PCDATA)>
<!ELEMENT CfgEntry(#PCDATA)>

```
<!--  This demonstrates an entry in a designated file for software configuration in the workaround to wipe off the vulnerability-->
<!ELEMENT WorkUrl (#PCDATA)>
<!--This is the workaound with a few URLs pointing to detailed information-->

<!ELEMENT Patch (PatchName, PatchNote, PatchTrace, PatchUrl*)>
<!ELEMENT PatchName (#PCDATA)>
<!ELEMENT PatchNote (#PCDATA)>
<!ELEMENT PatchTrace(RegistryTrc*, FileTrc*)>
<!ELEMENT RegistryTrc (TrcKey, TrcName, Trctype, TrcData)>
<!ELEMENT TrcKey(#PCDATA)>
<!ELEMENT TrcName(#PCDATA)>
<!ELEMENT TrcType(#PCDATA)>
<!ELEMENT TrcData(#PCDATA)>
<!--  This indicates the registry entry for check for the existence of patch-->
<!ELEMENT FileTrc (TrcPath, TrcName, TrcEntry)>
<!ELEMENT TrcPath (#PCDATA)>
<!ELEMENT TrcName (#PCDATA)>
<!ELEMENT TrcEntry (#PCDATA)>
<!--  This indicates the file entry for check for the existence of patch-->
<!ELEMENT PatchUrl (#PCDATA)>
<!--This gives the patch and related download sites. -->


<!ELEMENT Exploit (Precondition, Procedure, Post-condition)>

<!ELEMENT Precondition (PrivilegeNeed, PreConcept)>
<!ELEMENT PrivilegeNeed (#PCDATA)>
<!ATTLIST  PrivilegeNeed  language CDATA#REQUIRED>
<!ELEMENT PreConcept (#PCDATA)>
<!ATTLIST PreConcept  language CDATA#REQUIRED>
<!--  This gives the privilege needed for exploit in any language and high-level concept in any language. -->

<!ELEMENT Procedure(#PCDATA)>
<!ATTLIST  Procedure language CDATA#REQUIRED>
<!--  This is the procedure of exploit in source code or any other form of certain language-->

<!ELEMENT PostCondition(PrivilegeGot, PostConcept)>
<!ELEMENT PrivilegeGot(#PCDATA)>
<!ATTLIST  PrivilegeGot language CDATA#REQUIRED>
<!ELEMENT PostConcept(#PCDATA)>
<!ATTLIST  PostConcept  language CDATA#REQUIRED>
```

<!-- This gives the privilege got and high-level conceptual result of the exploit in any language. -->

<!ELEMENT VNote (#PCDATA)>
<!-- This is additional note of the vulnerability.-->
<!ELEMENT Credit (#PCDATA)>
<!-- This gives the acknowledgment-->

5 Application of CVML

CVML describes security vulnerabilities using XML in a structured, machine-readable way and provides a new security interoperability standard. It can be widely used in vulnerability research and security management of computer systems. We have implemented a prototype of automatic vulnerability management architecture shown in Fig.3.

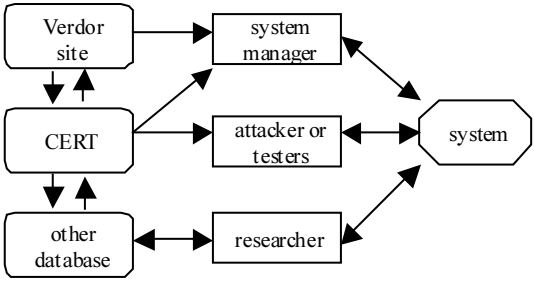

Fig. 3. Vulnerability Management with CVML

With CVML we can storage structured vulnerability information in database. Better analysis and data mining of vulnerabilities (for instance based on the element classification, evaluation etc) in individual database are provided. In our prototype, the combining component of database can collect, compare and combine vulnerability information from any sources (such as other databases, online submission etc.) in CVML, therefore leading to timely share of vulnerability knowledge and a large distributed database on the internet. The answering component of the database answers to a standard request from any sources (such as System Manager, Attack Tools, Assess Tools etc.) with related information in CVML. The subscription provider component will send messages in CVML about new vulnerabilities relevant to certain platform or software to the clients as soon as possible. Thus all kinds of security tools compatible with CVML could interact with vulnerability databases conveniently to support security scan or test.

According to a survey by SecurityFocus [7] system security administrators spend an average of 2.1 hours/day hunting for security information relevant in all kinds of security bulletins and mailing lists, then download and install patches to fix the sys-

tem. It is important to make this procedure automatic. In our prototype, System Manager is in charge of the security administration of one or more computer systems (usually in a local network). Ideally we expect in future all software installed on the system should register in the manager with the security related information (such as the version, the configuration, the update or patch sites etc). Under current circumstance system manager will scan the system periodically to get the related information of all software installed, the scan result is put into a XML file called System Description. Then system manager can use assess tools or assess engine in itself to check if any vulnerability exists using the CheckExist, Workaround, and Patch in CVML. The assess engine could maintain locally a cache of vulnerabilities relevant to its scope of usage. It also could inquire or subscribe for related vulnerabilities from certain databases. According to rules made by system administrators, System Manager will take some action (such as downloading and installing the patch automatically or only warning).

The attack tools or attack engine can be used for testing the security of systems or performing real attacks. It is based on the System Description (for test) or other system scan results (for real attacks). We have implemented in the prototype the privilege chaining process and some concept-level ratiocination of known vulnerabilities for automatic attack generation mainly using the element Exploit in CVML. Its interaction with vulnerability databases is similar to assess tools or assess engine.

We can see in our vulnerability management architecture, the component system manager scans, assesses, and tests the system periodically, update vulnerable software automatically when possible. This self-managing of systems not only alleviates the heavy burden of security administrators, but also improves the security of system remarkably.

6 Conclusions and Future Work

Until now, discovering, disclosing and patching vulnerabilities efficiently in information systems play the central role in security area. Current vulnerability information from different sources is mainly ambiguous text-based description that is not machine-readable and can't be efficiently shared, data-mined, combined or other in-depth automatic process. This paper presents a common vulnerability markup language (CVML) based on XML that describes vulnerabilities in a more structural way to support unified expression, easy sharing, automated management of vulnerability information. There are mainly 4 parts in CVML: General information, how to Check Existence, Solutions, and how to Exploit. In CVML, vulnerabilities can be described precisely from the view of both attackers and defenders. Architecture of automated vulnerability management based on CVML is also proposed. In this architecture, more manageable vulnerability databases are built; promulgating and sharing of vulnerability knowledge are easier; compare, and fusion of vulnerability information from different sources are more efficient; moreover automated scanning and patching of vulnerabilities leads to self-managing systems.

It is obviously not precise enough to check the existence of a vulnerability only based on the name and version of software installed. How can we do more efficiently? The granularity of privilege chaining is too big, how to improve it? How to improve the concept-level ratiocination of attack generation? How to improve the interaction between the concept-level and the functionality-level? These problems are in the scope of our future work.

References

1. IEEE.: The IEEE Standard Dictionary of Electrical and Electronics Terms. Sixth Edition. Institute of Electrical and Electronics Engineers Inc., New York NY (1996) 373
2. Amoroso, E.G.: Fundamentals of Computer Security Technology. Prentice-Hall PTR, Upper Saddle River NJ (1994)
3. John, D.H., Thomas, A.L.: A Common Language for Computer Security Incidents. Sandia Report, Sand98-8667. Livermore CA USA (1998)
4. http://cve.mitre.org
5. http://oval.mitre.org
6. http://www.owasp.org/vulnxml
7. http://www.securityfocus.com
8. http://www.cert.org
9. Carl, L.: A Taxonomy of Computer Program Security Flaws. Technical Report. Naval Research Laboratory (1993)
10. Brian, M.: A Survey of Software Fault Surveys. Technical Report, UIUCDCS-R-90-1651. University of Illinios at Urbana-Champaign (1990)
11. Taimur, A. (ed.): Use of A Taxonomy of Security Faults. Technical Report, TR96-051. COAST Laboratory, Department of Computer Sciences, Purdue University (1996)
12. Eckmann, S., Vigna, G., Kemmerer, R.: STATL. Technical Report, UCSB (2000)
13. Cuppens, F., Ortalo, R.: Lambda: A Language to Model a Database for Detection of Attacks. In: Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)
14. http: //www.insecure.org/nmap
15. http:// www.ietf.org/ids.by.wg/idwg.html

Trust on Web Browser: Attack vs. Defense

Tie-Yan Li and Yongdong Wu

Infocomm Security Department
Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
{litieyan, wydong}@i2r.a-star.edu.sg

Abstract. This paper proposes a browser spoofing attack which can break the weakest link from the server to user, i.e., man-computer-interface, and hence defeat the whole security system of Internet transaction. In this attack, when a client is misled to an attacker's site, or an attacker hijacks a connection, a set of malicious HTML files are downloaded to the client's machine. The files are used to create a spoofed browser including a faked window with malicious event processing methods. The bogus window, having the same appearance as the original one, shows the "good" web content with "bad" activities behind such as disclosing password stealthily. Once the attack is mounted, even a scrupulous user will trust the browser that is fully controlled by the attacker. We further propose several countermeasures against the attack.

1 Introduction

With the rapid development of Internet and web technologies, most of the online applications are built on or assisted by WWW. As shown in Figure 1, a typical web-based transaction consists of several components linked together for serving requests. Being requested, the web server receives a piece of reply data from application database. The web server then sends the data passing through the firewall, routers towards the requesting machine. Finally, the data is shown on the screen-user interface. We denote the secure link from the server to the user as *trust path*. Along the long link, various malicious attacks can be launched. For instance, the web site may suffer from database attack, web attack, and DDoS attack; the network may be assaulted by attacks such as eavesdropping, session hijacking, and routing disruption; virus may also disclose the information in the user's machine.

Because the important data may be eavesdropped, replayed or modified, security is the most important concern as of online transaction. The web site has a lot of security policies and means to guard the transaction system. Firewalls protect the hosts from malicious outside attacks; Intrusion Detection Systems monitor the networks and hosts all the time; Secure channels are built with security protocols (e.g., SSL [3]); And the host is well protected from virus. After all the defense are in place, are the shown data at the recipient side trustworthy? Most of the users will say "Yes". Why? The browsers produced by big players

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 241–253, 2003.

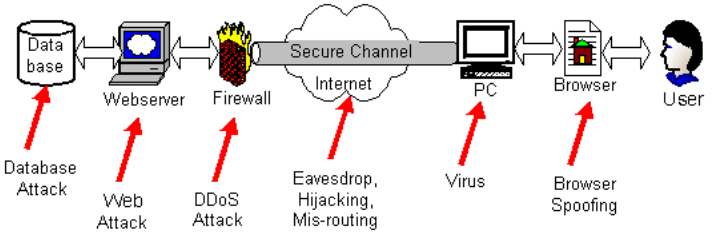

Fig. 1. Trust path

Microsoft or Navigator show the security sign on the screen. Regretfully, the answer is "No" because the security signs used for man-machine-interface can be easily reproduced with "Web spoofing" technique ([4] [5] [6] [7]). Therefore, the whole trust path can be easily broken at this weakest point by web spoofing. This topic is also mentioned in [2], which established several principles for designing secure user interface.

In this paper, we propose a browser spoofing attack simply using applet and frame to demonstrate that the popular commercial browsers (Microsoft Internet Explorer and Netscape Navigator) are not trustworthy. Comparing to web spoofing, browser spoofing can simulate dynamic event response as well as static visual appearance. To this end, the attacker lures the client to accept a faked HTML page at first. This HTML file is used to create a new window with a method `window.open()`. The bogus window, having the same appearance as the original browser window, shows the web content of the target server. By appending Java applet methods, the bogus browser can respond to the client's input events without being suspected. This attack allows an adversary to observe and modify all web pages sent to the client's machine, and observes all information input by the client. This attack is effective even though the web sever is SSL-enabled. Further, a user is unable to detect the attack even if he checks the security features, such as the security lock. In sum, the bogus browser, including the bogus window and the methods, is almost the same as the genuine browser from the viewpoint of the user. Meanwhile, the countermeasures against this attack are also provided.

The organization of the paper is as follows. Section 2 analyzes relevant works. Section 3 introduces the generic attack scheme. Section 4 addresses our implementation. Section 5 proposes the countermeasures toward the attack. At last, we conclude the paper and point out the future directions.

2 Related Works

Of all security techniques against Internet attacks, SSL3.0 [3] is the *de facto* standard for end-to-end security and widely applied to do secure transactions

such as Internet banking. In the ISO/OSI network reference model, SSL is located in the transport layer. Thus, SSL provides message confidentiality and integrity in transport layer. The higher application layer uses SSL as the secure transport channel. However, SSL only authenticate the transport connection, it does not authenticate the content sent to the recipient. That is to say, SSL guarantees that the received message is authentic and confidential in the transmission, but it does not care about the message before or after transmission. In all, SSL protects the connection rather than the applications. In a web application, a security lock is normally shown at the bottom of a browser window if SSL is completed successfully. Users trust this sign of security and further on, conduct "secure transactions". Therefore, if an attacker manages to produce a security lock appearing in the right place of the user's screen, he will convince the user to trust a faked site.

Felten et al. [4] proposed a web-spoofing attack. In the scheme, an attacker stays between the client and the target site such that all web pages destined to the user's machine are routed toward the attacker's server. The attacker rewrites the web pages in such a way that the appearances of these pages do not change at all. On the client browser, the normal status and menu information bar are replaced by identical-looking components supplied by the attacker. The attack [4] can prevent HTML source examination by using JavaScript to hide the browser's menu bar, replacing it with a menu bar that looks just like the original one. To attack a SSL-enabled web server, the attacker sends the client a certificate of an innocent person to avoid punishment. Although the method is quite straightforward, it is hard to launch because the attacker has to obtain the corresponding private key of the innocent person. It is also easy to be detected since a cautious user can also detect this attack by checking the security properties of the server.

Lefranc and Naccache [5] described malicious applets that use Java's sophisticated graphic features to rectify the browser's padlock area and cover the address bar with a false https domain name. Mounting this attack is much simpler than [4] as it only demands the insertion of an applet in the attacker's web page. The attack was successfully tested on Netscape's Navigator. But the attack is not successful when it is tested on Microsoft's Internet Explorer because a warning message is added in the end of the popup window. To overcome this shortcoming, the authors suggested to patch an artificial image. However, a weird image may also alert the client that an attack is under way. Moreover, Horton et al [6] and Paoli et al [7] adopted the patch method.

Ye et al [11] [12] proposed a trusted path solution to defend against web spoofing. The authors set up the boundary of the browser window with different colors according to certain rules. They defined an internal reference window whose color is randomly changed. Any malicious web content will fail to control a browser window due to uncontrollable inset/outset attributes. Therefore, if a new pop-up window has a different color from that of the reference window, the user concludes that a web-spoof attack is under way. For the device of small screen (such as hand-held device), this countermeasure is impractical because it is inconvenient to open two windows and switch between the windows. Moreover,

the attacker can create a bogus reference window to overlap the original reference window so as to break the defense.

The countermeasures are not limited in these "inside, software based" references. Some approaches [13] [14] used "outside, hardware based" references as the security evidences or trusted devices. Burnside et al [13] deployed a trustworthy camera which takes authentic pictures used for comparison with those on a large screen. Based on visual cryptography techniques, Tuyls et al [14] used an additional "decryption display" as the reference for protect two-way communication across insecure devices. One former approach [8] indicated several design issues of using mobile user devices in legally significant applications.

3 Attack on Trust Path

Trust relationship can be established in many ways. Authentication is one of the most popular means in reality. It checks what the claimant knows (e.g password), what the claimant has (e.g., security token), or what the claimant is (e.g. fingerprint). In the application of Internet transaction based on SSL protocol, the browser shows a security sign (e.g. security lock) if the server knows the private key conforming to a certificate. The browser vendors (e.g., Microsoft) declare that the server site is authentic if the lock sign is lit. However, the security lock-like sign is not enough to be used to trust a server due to our attack addressed in the following.

3.1 Attack Model

In this section, we design a model to attack SSL-enabled web application (because a non-secure web server is easy to be spoofed [4]). Refer to Figure 2, four kinds of participants are involved in this model: user, client, server and attacker. The normal transaction process is as follows. The user visits a web server via a web browser (client) such as Netscape Navigator or Microsoft Internet Explorer (IE). The SSL-enabled server owns a certificate issued by a certificate authority. When a user requests a secure page, SSL protocol authenticates the server and generates a session key for the secure communication between the client and the server. Meanwhile, the security lock is lit in the client's browser status line if the server is authenticated. Additionally, if the user clicks the security lock, the security information such as server certificate information will be shown on the data area of the popup window.

To mount the browser spoofing attack, the attacker should lure the client to accept malicious packets so as to sit in the middle between the target server and client. To this end, the attacker starts the attack in three ways before forging a SSL session.

1. The attacker controls a router or proxy in the communication path. The request toward a secure web server will be re-routed to an attacker's server.

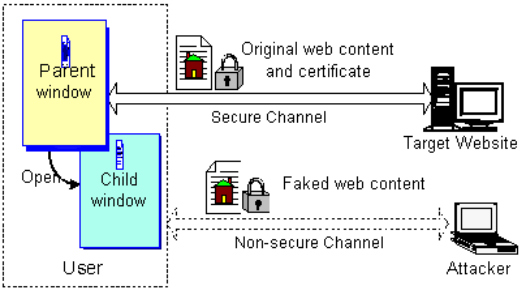

Fig. 2. Attack model

2. The attacker can hijack the communication session between the client and the server. When an initial SSL protocol request message is intercepted, a faked web page is sent to the client via non-SSL channel instead of the original one.
3. The client could be lured to browse the attacker's site pretended to be a trusted server. This method is often used since it can be done easily.

Thus, the attacker can insert, delete and tamper the communication data. S/he may feed the browser with the faked web content and a manipulated certificate. In sum, a malicious server is set up to communicate with the client "securely".

3.2 Spoofing Method

After forcing the client to receive malicious packets, the attacker sends a HTML file to the client so as to create a spoofed browser. This malicious HTML page may

1. create a new window with the method `window.open(attackerURL, "BogusWindow", "menubar=0, scrollbars=0, directories=0, resizable=0, toolbar=0, location=0, status=0")`, the parameter "0" indicates that the attacker disables the default browser window configuration.
2. draw a bogus status line and other GUI components with a security lock.
3. display the content of the target page and
4. create event response functions. For example, when the user checks the security property, an artificial dialog should be popped up to convince the user that all the security information is correct.

After creating the new window, there are two windows on the screen of the user's machine. One is original and the other is bogus. The spoofed one is in the front and is enlarged to overlap the original one. Although the spoofed window has the same interface as the genuine one, it is actually under the control of the attacker.

4 Implementation

Currently, two popular browsers are Microsoft Internet Explorer and Netscape Navigator. Our attack can be mounted on either of them. Because the implementations for both browsers are similar, we illustrate the attack to Internet banking between a SSL-enabled target site (Internet bank) and a user using Microsoft IE as a client. The target server has a certificate issued by some Certificate Authority (e.g., verisign.com) whose public key is embedded in Internet Explorer.

4.1 Spoofing the Browser Window

Most of the Internet surfers are familiar with the browser interface of Microsoft Internet Explorer, which looks like Figure 3. The user interface is a main window. In this window, its title bar includes a title and three system buttons. The following bar includes the menu bar and some shortcut icons. The address bar indicates the current web site, followed by the page content. In the last line, the status information is shown. The status information comprises of the icon of Internet Explorer logo, a security lock (a closed lock icon) representing that the present communication is secure, and some others.

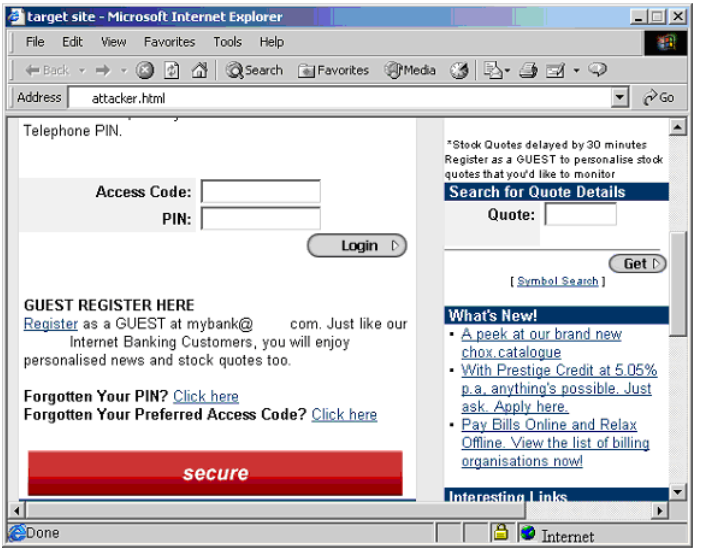

Fig. 3. Spoofed browser

Figure 3 is a logon page of an Internet banking system. It requires the bank customer to input a pair of account and password for identity verification. Here,

SSL protocol is executed and a security lock is shown on the status line. All the personal customer information are attempted to be encrypted with a session key and uploaded to the bank server. Unfortunately, the above interface is bogus and the personal information is sent to the attacker! The faked interface is created with our browser spoofing attack.

Technically, this bogus interface is generated after the client accepts a malicious HTML file (see Table 1) which creates a new window without status line. The new window is split into two frames. One is named as *upperFrame*. The *upperFrame* showing the web page `attacker.html` of the target site, is provided by the attacker. But from the view of the client, the content is not abnormal because the content is correct. The other frame is forged to be the status line of the genuine browser, and is split further into 5 sub-frames. Those sub-frames display IE icon with the HTML file `ielogo.html`, an earth icon with the HTML file `earth.html` and a security lock (closed) icon with the file `lock.html`.

Table 1. Frame settings

```
<FRAMESET ROWS="*,18" frameborder="YES" border="1"
                              framespacing="0">
   <FRAME NAME="upperFrame" SRC="attacker.html" >
   <FRAMESET cols="*,24,24,24,150" frameborder="YES" border="1"
                              framespacing="0">
      <frame name="ielogo" scrolling="NO" MARGINHEIGHT="0"
                              noresize src="ielogo.html">
      <frame name="status" scrolling="NO"MARGINHEIGHT="0"
                              noresize src="clearbg.html">
      <frame name="progress" scrolling="NO" MARGINHEIGHT="0"
                              noresize src="clearbg.html">
      <frame name="lock" scrolling="NO" MARGINHEIGHT="0"
                              noresize src="lock.html">
      <frame name="earth" scrolling="NO" MARGINHEIGHT="0"
                             noresize src="earth.html">
   </FRAMESET>
</FRAMESET>
```

To cheat the careful users, `lock.html` provides the security lock icon as well as its actions. Table 2 is the code of `lock.html` where `showLayer()` processes the click event. When the user clicks on the security lock icon, the response action will display the certificate information of the bank server as the genuine browser does.

4.2 Spoofing the Browser Methods

To enable the user to read the source of a web page, Microsoft IE provides "Viewing the Document Source" menu-item in the menu. In the web-spoofing attack proposed by Felten et al. [4], if the user chooses from the spoofed menu bar,

Table 2. Display security lock

```
<html>
    <head>
        <title>Untitled Document lock</title>
        <meta http-equiv="Content-Type" content="text/html;
                        charset=iso-8859-1">
    </head>
    <body bgcolor="#c8d0d4">
        <A onclick= "window.top.showLayer()"> <IMG src="lock.bmp"> </A>
    </body>
</html>
```

the attacker would display the original HTML source instead of the faked code. The attacker can also discourage the user from choosing the browser's "view document information" menu-item. Those attackers skilled in web development can further substitute the URL address field with a text field easily to forge the target server address. Thus, we do not implement the bogus menu-items, URL and the related methods. We focus on the security lock icon and the related mouse events. In a genuine browser, if a user clicks on the security lock so as to confirm the server further, the server information will be displayed on a movable popup window which is able to accept the user mouse input. To simulate the above functionalities for cheating the cautious user, the attacker creates the HTML tag <LAYER> other than the popup window to display the certificate information to avoid a warning message, while the response methods on mouse events are implemented in an applet.

Spoofing the Certificate Window. As mentioned in [9] [10], a HTML tag <LAYER> is a frame that can be absolutely positioned. It can occupy the same 2D spaces as another frame. A layer looks like a frame with a document property that is in turn an object, with all the properties of the top-level document object. It captures events in the same way as the top-level window or document. The basic properties are the same as the other HTML elements. The layer-specific attributes are "`top`", "`visibility`" and "`id`" properties: "`top`" property specifies its position so as to move the layer; "`visibility`" controls whether the layer is displayed;"`id`" identifies the layer. Table 3 sets up the layer to forge the certificate window. Table 3 is the code to generate the layer whose `id` is "*Layer1*", default visibility attribute is "*hidden*", and size is 400×500 pixels. Its top-left coordinate is (80, 40). The document loaded in this layer is the applet encapsulated in a jar file "`Spoof.jar`". This applet is enabled to communicate with the HTML JavaScript functions by setting the attribute "MAYSCRIPT=true". The code in Table 3 is appended to the server web, as well as the jar file "`Spoof.jar`".

As the normal browser, the applet shows the initial page of the certificate. As shown in Figure 4, the page includes 3 tabs: General, Detail and Certificate Path. The different tab page can be switched freely when the intended tab is clicked. Because the contents of these tabs are well known in advance, the attacker

Table 3. Embedding the spoof applet

```
<div id="Layer1" style="position:absolute; visibility:hidden; width:400px;
                        height:500px; z-index:1;left: 80px; top: 40px">
    <applet code=SpoofApplet archive="Spoof.jar" width=410 height=477
                        MAYSCRIPT=true>
    </applet>
</div>
```

encapsulates them in a jar file in advance so that the client can not detect the attack by checking the certificate, the expiry and any other information.

Fig. 4. Certificate window

Spoofing the Event Methods. The above step creates a static bogus certificate window which can cheat average users. To cheat careful users, the malicious applet should response to the user input dynamically. To this end, the bogus browser processes 5 kinds of events.

1. Click on the closed lock icon: The code in `lock.html` (see Table 2) indicates that the function `showLayer()` will make the layer visible when the *onClick* event occurs. The code in module `showLayer()` in Table 4 finds the document at first, then finds the layer based on layer `id` value "*Layer1*" attribute

which is defined in Table 3. The event response module changes the visibility attribute of the layer to be "*visible*" from "*hidden*". That is to say, the certificate window will be shown when the user clicks on the lock icon. In order to save time for next display of certificate information, the layer is hidden other than destroyed when closing certificate window.

Table 4. Change visibility of a layer

```
<Javascript>
Function showLayer()
        {var doc=window.top.frames[0].document;
         var layerstyle=doc.all["Layer1"]["style"];
         if( layerstyle.visibility=="hidden") layerstyle.visibility="visible";
         else layerstyle.visibility ="hidden"; }
</Javascript>
```

2. Click on the tab button: the corresponding tab page will be exposed. Class `JTabbedPane` provides an easy way to switch between the tabs.
3. Moving the certificate window: when the position of mouse is obtained, the new position of the layer can be set. However, the layer can move in the area of the bogus window only. This weakness may help a cautious user to detect the attack. Unfortunately, few users check the security by moving a window.
4. Click on the internal buttons: When a user clicks the buttons whose actions are restrained to the applet itself, the action procedures are easy to be realized. The exemplary button is the "Install Certificate ..." in the General tab page.
5. Click on external buttons: The buttons including the "OK" button and the "close" button in the top-right corner can close the certificate window. Because the button click event is received by the applet, the applet should pass it to the HTML/JavaScript page. By searching the `JSObject` tree, the method `actionListener` of `Okbutton` calls the Javascript function `showLayer()` in the HTML page. Table 5 illustrates this code.

Table 5. Response to click on lock

```
OkButton.addActionListener(new ActionListener() {
        Public void actionPerformed(ActionEvent e) {
                JSObject win=JSObject.getWindow(theApplet);
                        // theApplet is the name of the malicious applet.
                Object[] args=new Object[0];
                win.call("showLayer",args);}
});
```

6. When the user clicks on the system close button on the top-right corner of the browser window to close the window, the attribute of layer visibility is set to be "*hidden*".

5 Countermeasures toward Browser Spoofing

Our spoofing attack is very concise and effective. Any script kiddie can launch it easily. However, it is quite difficult to defend it. The reasons are

- Most users are not security expert. Their using behaviors are hard to be changed. If the users are used to trust a security lock, they will not be alerted by other anomaly events, e.g. changes on status bar.
- Even the expert themselves may be cheated by the faked lock and the faked certificates. Worst, if the terminal holding the browser is totally controlled by malicious environment, all tries of tracing a packet, redirect the connection to prove the original site would receive manipulated replies.

Actually, the failure mainly results from no clear visual sign to help the users' making a correct decision (i.e. a security lock is apparently not enough for application level security). To counter this attack, we have to rely on more visible and clear sign indicating certain trustworthy event. Two countermeasures are described in the following.

5.1 Disabling Configuration of Status Bar

In the browser spoofing attack, a new window is created with faked status line which indicates a bogus site to be authentic. To thwart this attack, the browser should disable status bar configuration when a new window is created. This modification is easy to implement without affecting the functionality of browsers. Thus, a security lock in status bar always shows the trustworthy site if and only if the server certificate is authentic. However, this countermeasure requires the user to pay attention to the status of the small security lock.

5.2 Synchronizing Colors

It is easier for a user to trust the web content if the colors are synchronized within a window. For example, dynamically controlling the color of a designated part of a window based on certain security rules. It is very hard for an attacker to synchronize the color without passing the security rules. Refer to Figure 5, the main menu bar, whose color is randomly set up by internal functions, is labelled as a standard reference. The boundary of the new generated windows will also show the same colors after accepting a certificate. If the colors are not identical, the new window can not be trusted.

Compared with [12], this solution requires no additional reference window. It could be ideal since it satisfies the necessary requirements in [12]: inclusiveness,

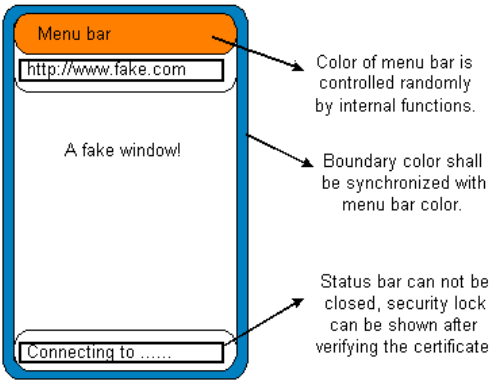

Fig. 5. Color synchronization

Effectiveness, no user work, no intrusiveness. It is also easy to distinguish the bad behavior windows. We believe that the browser spoofing attack is not incurable as long as the browser software providers patch the security hole as the above countermeasures.

6 Conclusions and Future Directions

In this paper, we demonstrated an effective attack - "browser spoofing" that make the browser un-trustable. We introduced a generic attacking scheme and elaborated its implementation. While using browser spoofing is not novel, the visual effect of this experiment can really cheat quite a lot of users. It shows that the trust path from user to web browser is weak, although security protocols like SSL are secure enough for end-to-end security. But the client side "end" security ends merely at transport layer. The gap still exists between the user and its browser, in which languages (i.e. Java, JavaScript) and dynamic properties (i.e. form functions, frames) provide rich effects, yet dangerous.

Although the attack is quite effective, it can be avoided by disabling the "status bar" and carefully detecting any anomaly happened there. In fact, to trust on the web browser, systematical defense technologies will be integrated together. The more complicated the strategies, the more user involvement. The less possible the attackers' following up, the more trustable the content. Hence, the challenge is how to balance the tradeoff between trust and ease of use.

Further on, we will study several secure schemes against the attacks for protecting web browser. We believe that the proposed attack is not incurable with effective trust scheme by a non-cautious user.

References

1. http://www.mymontage.com/.
2. Ka Ping Yee, User Interface Design for Secure System. ICICS, LNCS 2513, (2002)278–290
3. A. Freier, P. Kariton, P. Kocher, The SSL Protocol: Version 3.0.Netscape communications, Inc., (1996)
4. Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web spoofing: An Internet Con Game. 20th National Information Systems Security Conference,(1997) http://www.cs.princeton.edu/sip/pub/spoofing.html
5. Serge Lefranc and David Naccache, "Cut-&-Paste Attacks with Java". 5th International Conference on Information Security and Cryptology (ICISC 2002), LNCS 2587, pp.1–15, 2003.
6. Jeffrey Horton and Jennifer Seberry, Covert Distributed Computing Using Java Through Web Spoofing. ACISP (1998)48–57, http://www.uow.edu.au/ jennie/WEB/JavaDistComp.ps.
7. F. De Paoli, A.L. DosSantos and R.A. Kemmerer, Vulnerability of "Secure" Web Browsers. Proceedings of the National Information Systems Security Conference (1997)
8. Andreas Pftizmann, Birgit Pfitzmann, Matthias Schunter and Michael Waidner, Trusting Mobile User Devices and Security Modules, IEEE Computer, 30/2, Feb, 1997, p. 61–68.
9. Rick Darnell et al, Dynamic HTML. ISBN 0-57521-353-2(1998)
10. Gabriel Torok, Jeffrey Payne and Matt Weifeld, Javascript Primer Plus. ISBN 1-57169-041-7(1996)
11. Yougu Yuan, Eileen Zishuang Ye. Sean Smith, Web Spoofing. (2001) http://www.cs.dartmouth.edu/reports/abstracts/TR2001-409/
12. Eileen Zishuang Ye, Sean Smith, Trusted Paths for Browsers. 11th USENIX Security Symposium, (2002)
13. M. Burnside, Blaise Gassend, Thomas Kotwal, Matt Burnside, Marten van Dijk, Srinivas Devadas, and Ronald Rivest The untrusted computer problem and camera-based authentication. 1st International Conference on Pervasive Computing, LNCS 2414,(2002) 114–124.
14. Pim Tuyls, Tom Kevenaar, Geert-Jan Schrijen, Toine Staring, Marten van Dijk, Visual Crypto Displays enabling Secure Communications, Proceeding of First International Conference on Security in Pervasive Computing(2003)12–14

Security Protocols for Biometrics-Based Cardholder Authentication in Smartcards

Luciano Rila and Chris J. Mitchell

Information Security Group
Royal Holloway, University of London
Surrey, TW20 0EX, UK
{luciano.rila,c.mitchell}@rhul.ac.uk

Abstract. The use of biometrics, and fingerprint recognition in particular, for cardholder authentication in smartcard systems is growing in popularity, and such systems are the focus of this paper. In such a biometrics-based cardholder authentication system, sensitive data will typically need to be transferred between the smartcard and the card reader. We propose strategies to ensure integrity of the sensitive data exchanged between the smartcard and the card reader during authentication of the cardholder to the card, and also to provide mutual authentication between card and reader. We examine two possible types of attacks: replay attacks and active attacks in which an attacker is able to calculate hashes and modify messages accordingly.

Keywords: smartcards, biometrics, cardholder authentication.

1 Introduction

Traditional methods for automated personal identification mainly employ the possession of a token (magnetic card, USB-token) and/or the knowledge of a secret (password, PIN) to establish the identity of an individual. However a token can be lost, stolen, misplaced, or willingly given to an unauthorised person, and a secret can be forgotten, guessed, or unwillingly — or willingly — disclosed to an unauthorised person. Biometrics has emerged as a powerful tool for automated identification systems. Since it is based on physiological and behavioural characteristics of the individual, biometrics does not suffer from the disadvantages of the traditional methods.

In parallel, smartcards have steadily grown in popularity, as have their storage capacity and processing capabilities. The computing power of modern smartcards allows a wide variety of applications, from support for PKI to decentralised applications requiring off-line transactions [4,14,15]. Smartcards also offer the possibility of executing multiple applications on a single card. To implement controlled access to the functionalities of the card, smartcard systems typically require a method for cardholder authentication. Not only does cardholder authentication address the issue of card theft or misappropriation but it also allows the system to grant different access rights to different users of the same card.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 254–264, 2003.

An example of the latter can be drawn from health care applications where the patient and the physician access the same card belonging to the patient.

Biometrics and smartcards have the potential to be a very useful combination of technologies. On the one hand the security and convenience of biometrics allow for the implementation of high-security applications on smartcards. On the other hand smartcards represent a secure and portable way of storing biometric templates, which would otherwise need to be stored in a central database. Among the various biometrics technologies in use today, fingerprint recognition seems to be particularly suitable for smartcard systems.

A smartcard system is composed of two main physical units: the smartcard itself and the card reader. Depending on how the logical modules of the biometric system are distributed between the card and the card reader, biometrics-based cardholder authentication may require the transmission of sensitive data, such as a biometric live sample or a biometric template, between the two units. It is therefore fundamental to ensure the integrity of transmitted data during cardholder authentication. It is also important to provide mutual authentication between the two units, so as to prevent use of fraudulent cards or card readers.

In [3], weaknesses of the biometric system model are identified and some countermeasures are suggested, although no particular security protocol is proposed. Moreover, no assumptions as to the actual architecture of the system is made and the analysis is rather general in nature. In this paper, we analyse a specific system architecture that reflects the current state of the art for smartcard systems. Our focus is on security issues associated with the communications link between the smartcard and the card reader during fingerprint-based cardholder authentication. We propose strategies to ensure integrity of the data exchanged between the smartcard and the card reader, and also to provide mutual authentication between the two components. We examine two possible attacks: replay attacks, and active attacks in which an attacker is able to make minor modifications to the messages.

For the purposes of this analysis we do not make any assumptions about encryption or other cryptographic protection of the card/card reader communications link. Note that, in many cases, the requirements for high speed data transfer across this link, combined with the limited computational capabilities of the card, may severely limit the possibilities for such protection. We also assume throughout that fingerprint recognition is used as a method of cardholder authentication to the smartcard.

Given our focus on card/reader communications link, we make other simplifying assumptions. We assume that the smartcard is a tamper-proof device and any transmission between biometric system modules taking place within the card is therefore secure. We do not discuss the impact of using fake biometrics, such as plastic fingers, to fool the system, although it was shown in [17] that this is a possible attack with the current technology. This is an issue relating to the fingerprint-based biometric technology itself, and as such is outside the scope of this paper.

This paper is organised as follows. In Section 2 we define the system architecture analysed in this paper. In Section 3 we identify the possible threats to the communications link and propose two security protocols to prevent replay attacks. In Section 4, we discuss active attacks and propose security protocols to prevent them. We also propose a security protocol for mutual authentication of the smartcard and the card reader and analyse some methods for sharing of a secret key between the card and the reader. Finally we present our conclusions in Section 5.

2 Biometric System Architecture

2.1 General Model for Biometric Authentication

According to [5], a general biometric system is composed of the following logical modules:

1. Data collection subsystem;
2. Signal processing subsystem;
3. Matching subsystem;
4. Storage subsystem;
5. Decision subsystem;
6. Transmission subsystem.

A block diagram for the general authentication model is given in Figure 1. The authentication process involves the raw biometric data of the claimant being captured in the data collection subsystem by an input device or sensor, and then transferred to the signal processing subsystem where the feature extraction takes place. The matching subsystem receives the extracted features from the signal processing subsystem and retrieves the biometric reference template associated with the claimed identity from the storage subsystem. The matching subsystem then compares the submitted biometric sample with the reference template yielding a score, which is a numeric value indicating the degree of similarity between the submitted sample and the reference template. The decision subsystem receives the score and, according to a confidence value based on security risks and risk policy, decides whether to accept the claimant or not. The authentication decision is finally passed on to the application.

Note that these are logical modules, and therefore some systems may integrate several of these components into one physical unit.

2.2 Biometric System Architecture in a Smartcard System

The architecture of the biometric system within a smartcard system, i.e. how the logical modules of the biometric system are distributed between the smartcard and the card reader, determines the nature of the data to be transferred between the card and the card reader during biometric-based cardholder authentication. Hence different architectures are open to different threats. If all modules of the

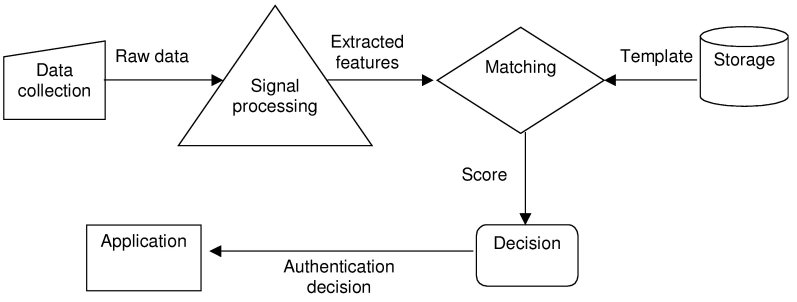

Fig. 1. General model for biometric authentication.

biometric system reside in the card, the whole cardholder authentication process takes place on the card and therefore no data transfer is needed. Provided that the card is a secure physical entity, this is the most secure architecture possible. However it is also the most costly and, with current technology, it is still a challenge to manufacture an ISO-compliant card containing all the modules of the biometric system.

Current biometrics-based smartcard systems must therefore distribute the modules of the biometric scheme between the card and the card reader in some way. In general, the signal processing module is very likely to be located in the card reader, since it requires both significant computational power and, most significantly, large amounts of RAM, and these requirements are likely to be beyond the capabilities of current smartcards. The location of the other modules, however, varies according to the system design and the biometric technology being used.

For fingerprint recognition, a recent prototype implemented in the Finger_Card project [18,13] has incorporated all but the signal processing module of the biometric system into the smartcard. Within such an architecture, the biometrics-based cardholder authentication is carried out as follows. The fingerprint image is collected by a sensor on the smartcard and transferred from card to reader. The reader performs feature extraction only, and transfers the extracted features back to the card. The card then performs the matching process and reaches the authentication decision. This architecture is represented in Figure 2 and is the basis of our analysis throughout the remainder of this paper.

3 Replay Attacks

In this paper we focus on possible attacks to the biometrics-based cardholder authentication process. For the purpose of this analysis, we assume that the smartcard is a tamper-proof device, and hence trust the security of any transmissions between those modules of the biometric system within the card. We therefore assume that the only communications link open to attacks is that be-

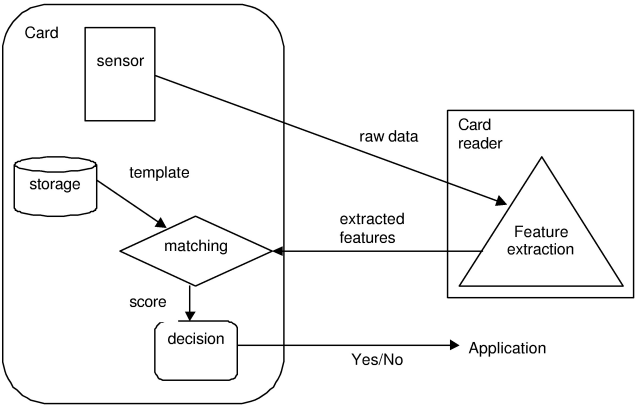

Fig. 2. The architecture of the biometrics-based authentication system within a smartcard system.

tween the smartcard and the card reader. The main threats to this link can be divided into threats to the 'up-link' (i.e. smartcard to reader) and 'down-link' (i.e. reader to smartcard).

In this section we assume that a possible attacker can only listen to the communications channel, record messages, and/or subvert the protocol by replaying old messages, but the attacker is unable to modify messages or create new messages. This is known as a replay attack [14]. Challenge-response protocols [9,10, 11] and zero-knowledge based protocols [12] may be used to counteract replay attacks.

3.1 Possible Threats

Suppose that an attacker is able to insert a bug in the communications link so that he can record the messages being transferred, and he can also intercept messages and replace them with old recorded messages. According to [16], the main threats to the communications link are as follows:

1. Up-link threats: an attacker can record the fingerprint image sent from card to reader while the legitimate cardholder is authenticating himself to the card. Once the attacker is in possession of a legitimate fingerprint image, he can steal the card, insert the card into the card reader, intercept whatever fingerprint image (probably his own) is being transferred to the card reader, and replace it with the legitimate fingerprint image. The card reader would then extract the features from a legitimate fingerprint image, and send them to the card which would most probably authenticate the user (an attacker) to the card.
2. Down-link threats: another possibility would be to record the (genuine) extracted features sent via the down-link, when the legitimate cardholder is

authenticating himself to the card. As before, the attacker can steal the card once in possession of a set of extracted features for the legitimate cardholder. When required to authenticate himself to the card, he can use his own fingerprint to enable the protocol to proceed. The card reader will extract the features from his illegitimate fingerprint image, which the attacker then intercepts and replaces with the legitimate set of features when they are sent across the down-link so that the attacker is authenticated to the card.

Note again that we are assuming here that the attacker can neither modify the messages nor create them, only replay them. Note also that both these attacks require the attacker to monitor card/reader communications when the card is in use by its legitimate holder, prior to stealing or otherwise misappropriating the card.

In the next two sections we propose two different security protocols designed to address the threats we have just described.

3.2 Security Protocol Using a Random String Generator on the Card

In this protocol, the freshness of the data is ensured by the use of random strings and hash functions. The following protocol prevents both replay attacks.

1. The reader generates a random string R_1 and sends it to the card.
2. The card (in fact the fingerprint sensor on the card) captures the fingerprint image ($BioData$), generates a random string R_2, and sends to the reader: $R_2 \mid\mid BioData \mid\mid h(R_2 \mid\mid R_1 \mid\mid BioData)$. Note that h denotes a cryptographic hash-function (see, for example, [14,6]), and $\mid\mid$ denotes concatenation of data items.
3. The reader computes $h(R_2 \mid\mid R_1 \mid\mid BioData)$ and verifies that it is identical to the value sent by the card.
4. The reader extracts the features (EF) from the fingerprint image and sends to the card: $EF \mid\mid h(R_1 \mid\mid R_2 \mid\mid EF)$.
5. The card computes $h(R_1 \mid\mid R_2 \mid\mid EF)$ and verifies that it is identical to the value sent by the reader.

Since the reader generates a new random string R_1 for each transaction, if the value $h(R_2 \mid\mid R_1 \mid\mid BioData)$ received by the reader is correct, then the message from the card is not a replay of an old message. Therefore steps (1), (2), and (3) prevent the up-link threat.

The down-link threat is prevented in steps (4) and (5). Since the card generates a new random string R_2 for each transaction, if the value $h(R_1 \mid\mid R_2 \mid\mid EF)$ received by the card verifies correctly, then the message from the reader to the card is not a replay of an old message.

Note that this protocol also partially ensures the integrity of the data being transferred in both directions even though integrity is not an issue when replay attacks are considered. In the up-link, any change in $BioData$ during transmission would result in an incorrect hash-code $h(R_2 \mid\mid R_1 \mid\mid BioData)$ in step 3. In

the down-link, any change in EF during transmission would result in an incorrect hash-code $h(R_1 \,||\, R_2 \,||\, EF)$ in step 5. Of course, a sophisticated attacker could change the data and also recompute the hash-value; however, such an attack requires a level of sophistication beyond the scope of the countermeasures considered in this section.

3.3 Security Protocol Using Biometric Data as a Random String

The nature of biometric data is such that two different measurements of the same biometric feature from the same person are very likely to be different, although the difference may be small. For example, when fingerprint recognition is used, the fingerprint image captured by the sensor may vary, e.g., due to skin conditions, dirt, grease, or the position in which the finger is placed on the sensor. The biometric data can therefore be regarded as a unique random string for each transaction, and can as such be incorporated by the card into the security protocol. Although this approach may restrict the possible values for the random string, it may have practical advantages if generating a random string is a demanding task for the smartcard.

The protocol presented in the previous section would then be modified as follows:

1. The reader generates a random string R and sends it to the card.
2. The card captures the fingerprint image ($BioData$) and sends to the reader: $BioData \,||\, h(BioData \,||\, R)$.
3. The reader computes $h(BioData \,||\, R)$ and verifies that it is identical to the value sent by the card.
4. The reader extracts the features (EF) from the fingerprint image and sends to the card: $EF \,||\, h(EF \,||\, h(BioData \,||\, R))$.
5. The card computes $h(EF \,||\, h(BioData \,||\, R))$ and verifies that it is identical to the value sent by the reader.

As in the previous section, a new random string R is generated by the reader for each transaction, and the use of the hash-code $h(BioData \,||\, R)$ prevents replay attacks in the up-link — steps (1), (2), and (3).

The down-link threat is prevented in steps (4) and (5). Under the assumption that the biometric data is very likely to be different at every collection, $BioData$ plays the role of a random string generated by the card. When the reader returns to the card the value $h(EF \,||\, h(BioData \,||\, R))$ in step (4), the verification of the hash-code in step (5) ensures that this is not a replay of an old message.

Note again that this protocol also partially ensures the integrity of the data being transferred in both directions.

4 Active Attacks

Suppose now that the attacker can modify messages and also knows the details of the protocol being used (including the hash-function). Since we have assumed

that all data (including the random strings) are transferred in plaintext, the protocols of Section 3 can be defeated, since the attacker could intercept either the fingerprint image or the extracted features and simply generate the hashes as the protocol goes along. The integrity of the data being transferred between the card and the card reader becomes an issue in this case. In order to prevent such active attacks, we describe a slightly different protocol.

4.1 Message Authentication Codes (MACs) and Mutual Authentication

Suppose the card and the card reader share a secret key. This allows both parties to use a Message Authentication Code (MAC) in the place of the hash-code in the protocols of Section 3. A MAC function, as specified in [14,7,8], is a key-dependent cryptographic function with the property that it can map arbitrarily long messages to fixed length strings. Moreover, only someone who knows the key can produce a valid MAC for a data string. MACs are validated by re-computing them using the shared secret key.

Hence, given that the secret key shared by card and card reader is not known to the attacker, only a legitimate card and card reader can generate and verify valid MACs. The protocols described in Section 3 remain the same except that MACs are used instead of one-way hash-functions, and the protocols now provide mutual authentication between the card and the card reader (see, for example, [14,11]).

4.2 Mutual Authentication Protocol Prior to Transaction

The sharing of a secret key would also make it possible for the card and the card reader to run a mutual authentication protocol before any sensitive data is transferred. In this way the card would be assured that it has not been inserted into a hostile card reader. Conversely, the card reader can verify that the card inserted is a legitimate one. A possible protocol for this scenario is as follows. This protocol actually conforms to the relevant ISO/IEC standard for authentication protocols, [9], and is closely related to the SKID3 protocol, [1]. Note that in this protocol m denotes a MAC function; specifically, $m_K(X)$ denotes the MAC computed on data string X using key K.

1. The reader generates a random string R_1 and sends it to the card.
2. The card generates a random string R_2 and sends to the reader the two values: $R_2 \,||\, m_K(R_2 \,||\, R_1)$.
3. The reader computes $m_K(R_2 \,||\, R_1)$ and verifies that it is identical to the value sent by the card.
4. The reader sends to the card: $m_K(R_1 \,||\, R_2)$.
5. The card computes $m_K(R_1 \,||\, R_2)$ and verifies that it is identical to the value sent by the reader.

One possible reason for taking this approach — rather than the one in Section 4.1 — might be that computing a MAC on the fingerprint data is not feasible. This is because, whereas PINs are very short, fingerprint samples are rather large, and the limited computational and storage capabilities of the card may severely limit the possibilities of a MAC computation on the fingerprint data. Hence conducting the authentication protocol before the exchange of sensitive data would be a reasonable alternative. Note however that this approach does not prevent active attacks such as those described at the beginning of Section 4.

4.3 Sharing a Secret Key

Methods by which the card and the card reader can share a secret key are now considered. One possible way to accomplish this is to have a single key shared by all cards and card readers in the application. The fact that a universal key is shared by all parties in the application domain is a potential source of weakness, but the use of a secret key nevertheless makes things more difficult for an attacker. Measures could also be adopted to improve the overall security of such a system.

One such measure is to have expiry dates for keys. Keys would expire periodically and new keys would be generated — together with a new card possibly. Card readers would have to be equipped with copies of all the current keys. If a key is compromised, this measure would at least limit the period of time in which attacks could be carried out. A second approach would have all parties in the application sharing a list of keys (K_1, K_2, ..., K_n say) rather than a single key. At the beginning of the protocol, in step 2, the card chooses at random a key K_i ($1 < i < n$) from the list, and sends to the card reader:

$$R_2 \,||\, Biodata \,||\, i \,||\, m_{K_i}(R_2 \,||\, R_1 \,||\, Biodata).$$

The card reader uses the key index i to determine which key is being used. The protocol then proceeds with all MACs generated using K_i. In this case, even if one key is compromised (e.g. by cryptanalytic means), the attacker would have to make many attempts until the compromised key is used again. However, after a few failed attempts to use the card, the card would be blocked, making it unlikely that this attack would work. In order to further improve security, cards could be given different subsets of the same set of keys, although all readers would have all possible keys.

A third approach avoids the weakness of using a universal key by giving a unique key to each card. Each card reader would then need to have all such keys or, at least, on-line access to a centralised database of keys. The need for a key database could be avoided by deriving all card keys from a single secret master key. Specifically it would be possible to derive a card secret key from a combination of a secret master key with a card serial number (e.g. using a one-way function such as a cryptographic hash-function exhibiting pseudorandomness properties). Such a scheme will work as long as the card reader has access to the master key. Card key derivation from a card issuer secret master key is a solution

described in Annex A1.4 of EMV Book 2 [2], an industry standard governing interactions between a card and terminal, and used for smartcard-based debit and credit cards.

5 Conclusions

All the threats examined in this paper arise from an assumed lack of integrity for the communications link between card and card reader. If it is assumed that the card reader is a trusted device and has not been interfered with or replaced, then guaranteeing the integrity of the link between the card and the card reader would effectively prevent all the threats, even in the absence of any confidentiality for data transferred.

If the integrity of the communications link is untrusted, then security protocols are able to prevent some attacks. Replay attacks can be prevented using random numbers and hash-functions to ensure the freshness of the data. This countermeasure requires the card to be capable of computing hash-functions and/or generating random numbers. Alternatively, by its very randomic nature, the biometric data may play the role of the random string generated on the smartcard. Active attacks can be prevented using similar protocols, using MACs instead of hash-functions. MACs can also be used to provide mutual authentication between the card and the reader. If the cryptographic capability of the card is limited, then a mutual authentication protocol can be conducted before any sensitive data is exchanged, instead of protecting the actual data transfer. In any case, the use of MACs requires that the card and the reader share a secret key, and possible means by which such keys could be managed have also been discussed.

Acknowledgments. The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-2000-25168 (Finger_Card). The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

The authors would like to thank their colleagues in the Finger_Card Project for their encouragement and advice.

References

1. A. Bosselaers and B. Preneel (editors), *Integrity primitives for secure information systems: final report of RACE integrity primitives evaluation RIPE-RACE 1040*, LNCS 1007, Springer-Verlag, 1995.
2. EMV 2000, Integrated circuit card specification for payment systems, Book 2 — Security and key management, version 4.0, 2000.

3. G. Hachez, F. Koeune, and J.-J. Quisquater, "Biometrics, access control, smart cards: a not so simple combination", in *Proc. 4th Smart Card Research and Advanced Applications Conference (CARDIS 2000)*, J. Domingo-Ferrer, D. Chan, and A. Watson (editors), pp. 273–288, Kluwer Academic Publishers, Bristol, UK, Sep. 2000.
4. M. Hendry, *Smart Card Security and Applications*. Artech House, 1997.
5. ISO/DIS 21352: 2001, Biometric information management and security, ISO/IEC JTC 1/SC 27 N2949.
6. ISO/IEC 10118–3: 1998, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions.
7. ISO/IEC 9797–1: 1999, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher.
8. ISO/IEC 9797–2: 2002, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a dedicated hash-function.
9. ISO/IEC 9798–2: 1999, Information technology — Security techniques — Entity authentication — Part 2: Mechanisms using symmetric encipherment algorithms.
10. ISO/IEC 9798–3: 1998, Information technology — Security techniques — Entity authentication — Part 3: Mechanisms using digital signature techniques.
11. ISO/IEC 9798–4: 1999, Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function.
12. ISO/IEC 9798–5: 1999, Information technology — Security techniques — Entity authentication — Part 5: Mechanisms using zero knowledge techniques.
13. M. Janke, "Bio-System-On-Card", in *SecureCard 2001*, Jun. 2001, Hamburg, Germany.
14. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.
15. W. Rankl and W. Effing, *Smart Card Handbook*. John Wiley & Sons, 2001.
16. L. Rila and C. J. Mitchell, "Security analysis of smartcard to card reader communications for biometric cardholder authentication", in *Proc. 5th Smart Card Research and Advanced Application Conference (CARDIS '02)*, pp. 19–28, USENIX Association, San Jose, California, Nov. 2002.
17. T. van der Putte and J. Keuning, "Biometrical fingerprint recognition: don't get your fingers burned", in *Proc. 4th Smart Card Research and Advanced Applications Conference (CARDIS 2000)*, J. Domingo-Ferrer, D. Chan, and A. Watson (editors), pp. 273–288, Kluwer Academic Publishers, Bristol, UK, Sep. 2000.
18. B. Wirtz, "Biometric System On Card", in *Information Security Solutions Europe 2001*, Sept. 2001, London, UK.

Does It Need Trusted Third Party? Design of Buyer-Seller Watermarking Protocol without Trusted Third Party*

Jae-Gwi Choi[1], Kouichi Sakurai[2], and Ji-Hwan Park[1]

[1] Department of Information Security, Pukyong National Univ. 599-1 Daeyeon-dong Nam-ku Busan, Korea
jae@mail1.pknu.ac.kr, jpark@pknu.ac.kr

[2] Faculty of Information Science and Electrical Engineering, Kyushu Univ. 6-10-1 Hakozaki Higashi-ku Fukuoka, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. Buyer-seller watermarking protocol is a combination of traditional watermarking and fingerprinting techniques. For example, in applications where multimedia content is electronically distributed over a network, the content owner can embed a distinct watermark (a fingerprint), in each copy of the data that is distributed. If unauthorized copies of the data are found, then the origin of the copy can be determined by retrieving the unique watermark corresponding to each buyer. Recently, Ju and Kim proposed an anonymous buyer-seller watermarking protocol, where a buyer can purchase contents anonymously, but the anonymity can be controlled. They used two trusted parties: the watermark certification authority and the judge. The significance of this protocol is that it offered anonymity to watermarking protocol. But this protocol has the problem that honest buyers can be found as guilty, because sellers can recreate the same contents as the buyer's one if he/she colludes with the watermark certification authority and the judge. Thus this scheme must assume existence of the trusted third parties for its security. In this paper, we show shortcomings of this protocol and suggest a buyer-seller watermarking protocol that provides security of buyers and sellers without trusted third party.

1 Introduction

The current rapid development of new Information Technology and electronic commerce has resulted in a strong demand for reliable and secure copyright protection techniques for multimedia data. Over the past few decades, a considerable number of studies have been conducted on the design of methods that technically support the copyright protection of digital data. Copyright marking schemes have been proposed as the important class of these techniques. They are the embedding of marks into digital contents that can later be detected to identify owners (*watermarking*) or recipi-

* The first and third authors were partly supported by grant No.01-2002-000-00589-0 from the Basic Research Program of the Korea Science & Engineering Foundation and by University IT Research Center Project, MIC, Korea. It was done while the first author visits in Kyushu Univ. with the support of Association of International Education, Japan.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 265–279, 2003.

ents (*fingerprinting*) of the content. While digital fingerprinting schemes enable a seller to identify the buyer of an illegally distributed content by providing each buyer with a slightly different version, digital watermarking schemes enable the seller/content owner to prove the rights of the contents by embedding the seller's information into the contents. Buyer-seller watermarking protocol is a combination of traditional watermarking and fingerprinting techniques.

1.1 Related Works

***Symmetric Schemes*:** Classical fingerprinting and watermarking schemes [QN98][Ne83] are symmetrical in the sense that the content owner knows the watermarks uniquely linked with the buyer. Thus, if another copy with this watermark turns up, the buyer can claim that the seller redistributed it. Because this could be done for example, by a malicious seller who may want to gain money by wrongly claiming that there are illegal copies around. Thus, one cannot really assign responsibility about redistribution to one of them.

***Asymmetric Schemes*:** This problem is overcome by asymmetric schemes [PS96][MW01]. Here, because only the buyer can obtain the exact watermarked (fingerprinted) copy, he/she cannot claim that an unauthorized copy may have originated from the seller. Hence, if an unauthorized copy is found, the seller can obtain a means to prove to a third party that the buyer redistributed it and he/she can identify a traitor/copyright violator. However the drawback of these solutions is that it did not provide a buyer's anonymity.

***Anonymous Schemes*:** To protect buyer's privacy two anonymous schemes have been suggested by Pfitzman et al [PW97] and Ju and Kim [JK02]. The idea is that the seller can know neither the watermarked content nor the buyer's real identity. Nevertheless the seller can identify the copyright violator later. This possibility of identification will only exist for a copyright violator, whereas honest buyers will remain anonymous.

Requirements of anonymous buyer-seller watermarking protocols can be listed as follows [JK02][PW97]:

1. **Anonymity:** A buyer should be able to purchase digital contents anonymously.
2. **Unlinkability:** Given two digital contents, nobody can decide whether or not these two contents were purchased by the same buyer.
3. **Traceability:** The buyer who has distributed digital contents illegally (traitor/copyright violator) can be traced.
4. **No Framing (Buyer's security):** An honest buyer should not be falsely accused by a malicious seller or other buyers.
5. **No Repudiation (Seller's security):** The buyer accused of reselling an unauthorized copy should not be able to claim that the copy was created by the seller or a security breach of the seller's system.
6. **Collusion Tolerance:** Attacker should not be able to find, generate, or delete the fingerprint by comparing the copies, even if they have access to a certain number of copies.

[PW97] scheme is inefficient and impractical because it is based on secure two-party computations [CD87] (It use general theorems like "every NP-language has a zero-knowledge proof systems" without presenting explicit protocols) with high complexity. Later, [PS00] suggested an efficient method without secure two party computations. But this method is also impractical because it used [BS95] scheme as a building block for collusion resistance. In [BS95], their code needed for embedding is so long that the overall system cannot be practical. Recently, [JK02] proposed an anonymous buyer-seller watermarking protocol, adding anonymity and unlinkability to [MW01] scheme. [JK02] scheme and [MW01] scheme used Cox's invisible watermarking algorithm [CK97] as a building block. [JK02] scheme is a significant model in the sense that it offered the anonymity of a buyer to watermarking protocol. But the problem of this protocol is that it cannot provide security of sellers and buyers, because a seller can recreate the buyer's copy if he/she colludes with the watermark certification authority and the judge.

1.2 Our Contributions

In this paper, we suggest secure buyer-seller watermarking protocol against conspiracy attack, which can solve the problem of [JK02] scheme. '*Conspiracy attack*' is means that a seller colludes with the watermark certification authority or the judge in order to recreate buyer's copy for his/her gain.

We compare the features of our proposal with [JK02], [MW01] in Table 1.

Table 1. Comparison of our proposal with [MW01] and [JK02]

Features	[MW01]	[JK02]	Our Proposal
Anonymity	No Offer	Offer	Offer
Unlinkability	No Offer	Offer	Offer
No two-party computation	Yes	Yes	Yes
No Framing	No Offer[1]	No Offer[2]	Offer
No Repudiation	No Offer[1]	No Offer[2]	Offer
Participators of Identification protocol	Arbiter, seller, Buyer	Judge, watermark certification authority, seller	Arbiter, watermark certification authority, seller, buyer

1: [MW01] scheme provides the pertinent function only if the watermark certification authority is memoryless and not malicious.

2: [JK02] scheme provides the pertinent function only if the watermark certification authority and the judge are not malicious.

The most meaningful feature of our scheme is that there is no need to assume the trusted third party (the watermark certification authority and the judge)'s honesty. On the contrary, [JK02] and [MW01] must assume that the watermark certification

authority does not collude with a seller or a buyer for security of their protocols, because the watermark certification authority knows the buyer's unique watermark. Besides, [JK02] scheme also must assume honesty of the judge, because a buyer's secret key is encrypted with the judge's public key. Since, the buyer's secret key is used in encryption of content and anonymity offering in [JK02] scheme, it must not be revealed. And, the judge of [JK02] scheme is not an arbitrator but the fixed party from the first stage (watermark generation protocol). On the other hand, the judge of our scheme and [MW01] scheme is a complete arbitrator because the judge is not involved in other protocols except identification protocol.

1.3 Our Approach

The main idea of our scheme is to use a commutative cryptosystems in watermark generation protocol in order to prevent conspiracy attack. In our scheme, the watermark certification authority issues the buyer's unique watermark to the buyer but he cannot know which watermark the buyer chose. Thus even if a seller colludes with the watermark certification authority, he cannot recreate the buyer's copy. The second idea is that buyers generate two secret keys by splitting the original secret key corresponding with her real identity. One is used in encryption of content and the other is used in her owns anonymity. In our scheme, the others except the buyer cannot know the buyer's secret key will be used in decryption. Thus the others cannot recreate the same watermarked contents as the buyer's one, if computing discrete logarithms is hard.

The rest of this paper is organized as follows. First, [JK02] scheme is described briefly and its shortcomings are discussed in Section 2. Next, cryptographic primitive is described in Section 3. Then, the proposed buyer-seller watermarking protocol is described in detail in Section 4 and various features of the proposed scheme are analyzed in Section 5. Finally, we conclude in Section 6.

2 Overview of the Attacked Scheme

In this section we briefly review the construction proposed in [JK02]. For simplicity we use the same notations.

2.1 Ju and Kim's Scheme

Preprocessing: All participants have a pair of a private key and a public key (sk, pk) certificated by certificate authority (CA).

Watermark Generation: A buyer generates an anonymous key pair of a private key and a public key (sk_B^*, pk_B^*). A buyer generates $C = E_{pk_j}(sk_B^*)$ and *cert* proving

that sk_B^* is a discrete logarithm or e-th root of a given pk_B^* without disclosing sk_B^* using a verifiable encryption scheme[1] (E). In here, pk_J is the public key of the judge. After the buyer transmits C, pk_B^*, signature of $pk_B^* : sign_{sk_B}(pk_B^*)$ and the certificate $cert$ to the watermark certification authority, the authority verifies the certificate. If it is verified, the watermark certification authority is convinced that C is indeed the encryption of sk_B^*. Then, the watermark certification authority generates a watermark $W = \{w_1, w_2, \square, w_n\}$ randomly and sends to the buyer the anonymous public key pk_B^* and the watermark encrypted with the buyer's anonymous public key $w = E_{pk_B^*}(W)$ along with $s = sign_{sk_w}(w \| pk_B^*)$, which certifies the validity of the watermark and also ensures that pk_B^* was used to encrypt W as public key. The watermark certification authority stores $B, w, s, pk_B^*, sign_{sk_B}(pk_B^*)$, and $(C, cert)$ in $Table_W$. Here $\|$ denotes a concatenation and the encryption algorithm is homomorphic[2].

Watermark Insertion: A buyer sends $pk_B^*, E_{pk_B^*}(W), s$ to the seller to obtain a watermarked content. By verifying the signature with the watermark certification authority's public key, the seller is convinced of the watermark's validity. If the verification holds, the seller generates a unique watermark V and embeds it into multimedia content X. Let X' be the watermarked content with V. To embed the second watermark W generated by the watermark certification authority into X' without decrypting $E_{pk_B^*}(W)$, the seller encrypts the watermarked content X' with pk_B^* and finds the permutation σ satisfying $\sigma(E_{pk_B^*}(W)) = E_{pk_B^*}(\sigma(W))$. Because of the homomorphic property of the encryption algorithm E used by the watermark certification authority, the seller can compute watermarked content $E_{pk_B^*}(X'')$. Where $\oplus$ denotes the embedding operation. The seller transmits the computed $E_{pk_B^*}(X'')$ to the buyer and stores pk_B^*, w, s, σ and V in his/her table $Table_B$.

$$E_{pk_B^*}(X'') = E_{pk_B^*}(X') \oplus \sigma(E_{pk_B^*}(W)) = E_{pk_B^*}(X \oplus V) \oplus E_{pk_B^*}(\sigma(W))$$
$$= E_{pk_B^*}(X \oplus V \oplus \sigma(W))$$

---

[1] The idea is that if A and B wish to exchange their signatures on some message, they will first exchange verifiable encryption of them, using as E the public key of some trusted third party. If this was successful, it will be safe for A to just reveal his signature to B. Even if B never answers, A can get B's signature by having the trusted party decrypt it [CD98].

[2] A public key encryption functions $E : G \to R$ defined on a group $(G, \cdot)$ is said to be homomorphic if E forms a homomorphism. That is, given $E(x)$ and $E(y)$ for some unknown $x, y \in G$, anyone can compute $E(x \cdot y)$ without any need for the secret key. In other words, by privacy homomorphism with respect to $\oplus$, it means it has the property that $E_{pk}(x \oplus y) = E_{pk}(x) \oplus E_{pk}(y)$.

Copyright Violator Identification: When an illegal copy Y of an original content X is discovered, the seller extracts the unique watermark U in Y using detection algorithm. Then, he/she finds the buyer's information $pk_B^*, E_{pk_B^*}(W), s, \sigma$ stored with V with the highest correlation by examining the correlations of extracted watermark U and all $V's$ in the $Table_B$. And the seller sends them with X, Y to the judge. The judge verifies $sign_{sk_B}(pk_B^*)$ and $cert$ with the help of the watermark certification authority, and recovers the buyer's anonymous private key sk_B^*. If the verification success, judge computes $\sigma(W)$ and checks the existence of $\sigma(W)$ in Y by extracting the watermark from Y and estimating its correlations with $\sigma(W)$. If there exists $\sigma(W)$, the buyer is guilty and the buyer's ID (B) is revealed to the seller.

2.2 Analysis of the Scheme

2.2.1 Observations on Security

This scheme is very efficient in the sense that identification protocol is carried out without any help of the accused buyer. But the most undesirable issue of [JK02] is that the seller can recreate the buyer's copy if he colludes with the watermark certification authority and the judge. Thus the seller can cheat an honest buyer in this scheme.

- **Conspiracy Attack I: Collusion of the Seller, the Watermark Certification Authority and the Judge**

  To forge illegal copy, Y with the special watermark W, first the seller sends pk_B^*, s received from a buyer to the watermark certification authority. The watermark certification authority searches for the buyer's information, $[w = E_{pk_B^*}(W), C = E_{pk_J}(sk_B^*), pk_B^*]$, corresponding with pk_B^*, s in $Table_W$ and sends it (C) to the judge. The judge decrypts C and sends sk_B^* to watermark certification authority. The watermark certification authority decrypt □ using sk_B^* received from the judge and sends it to the seller. Then, the seller can recreate the buyer's copy because he/she knows the buyer's unique watermark, W.

- **Conspiracy Attack II: Collusion of the Seller and the Judge**

  [JK02] insists that only the buyer can decrypt the watermarked contents, because the watermarked content $E_{pk_B^*}(X'')$ encrypted with the buyer's anonymous public key pk_B^*. However in this protocol, a seller can obtain $C = E_{pk_J}(sk_B^*), pk_B^*$ are transmitted through insecure channel in the watermark generation protocol. If a seller obtains C, pk_B^*, she/he researches the buyer's record corresponding with pk_B^* at $Table_B$ and sends $C, E_{pk_B^*}(X'')$ to

the judge. These are just plain text in the view of the judge. Thus the seller (or the judge) can decrypt the buyer's copy X''.

In this scheme, the seller cannot obtain proof of treachery, because the accused buyer can claim that the unauthorized copy was created by the seller. After all, [JK02] scheme is weak against conspiracy attack. Of course, [JK02] assumes that the watermark certification authority and the judge are TTP and do not collude with a seller. But in principle, in the model for anonymous protocol the trust in the authority should be minimal. Note that when talking about attackers we also mean collusions of sellers and watermark certification authority and the judge. In other words, the seller must be able to execute all processes securely without compromising her private key even if the attacker is a trust center.

2.2.2 Observations on Efficiency

[JK02] is based on public key encryption schemes with homomorphic property [MW01] and a verifiable encryption schemes [CD98]. It presupposes additional condition as existence of the secure verifiable encryption scheme compared with [MW01] ([MW01] is based on the public key encryption schemes with homomorphic property and Cox's algorithm [CK97]). In [JK02], a verifiable encryption scheme is used in order to carry out identification protocol without any help of the accused buyer. But, a verifiable encryption scheme must take some care needs such as the secure hash function etc., [CD98] to avoid that one party falsely accuses the other of cheating.

The next undesirable point is protection of the buyer's anonymity. Most of anonymous protocols minimize the possibility of a buyer's real ID's exposure using a method: the pseudonym of a buyer is only known to an authority (normally registration center - watermark certification authority in [JK02]). But in this protocol, both the watermark certification authority and the judge can know the buyer's real ID corresponding with the buyer's anonymous public key. Besides, the judge of [JK02] scheme should be restricted. The judge that buyers chose in the watermark generation protocol must take part in identification protocol in order to identify a copyright violator. In Comparison with other anonymous protocols (Whoever is honest can be an arbiter), [JK02] scheme is inefficient in this aspect.

3 Cryptographic Primitive

3.1 Commutative Cryptosystems

We introduce the commutative cryptosystem in order to prevent the collusion of the seller and the watermark certificate center. In our protocol, even if the watermark certification authority issues the buyer's unique watermark, he/she cannot know which watermark is the buyer's one. We briefly describe the commutative cryptosystems introduced by Zhao and Varadharajan [ZV03][3] in the following.

---

[3] Commutative cryptosystems are often used in mental poker game [GM82][ZV03]. The hard part of mental poker is dealing the cards. Hands must be random and disjoint, and players

There are two parties Alice and Bob and they use the same prime number p. They have

$$K_A = \{(p, \alpha_A, k_A, \beta_A) : \beta_A \equiv {\alpha_A}^{k_A} \pmod p\}$$
$$K_B = \{(p, \alpha_B, k_B, \beta_B) : \beta_B \equiv {\alpha_B}^{k_B} \pmod p\}$$

Encryption

The original message is x. Alice chooses random value number r_A, and the result of encryption with K_A has two parts y_{1A} and y_{2A}: $E_{K_A} = (y_{1A}, y_{2A})$.

$$y_{1A} \equiv {\alpha_A}^{r_A} \pmod p, y_{2A} \equiv x{\beta_A}^{r_A} \pmod p$$

Bob chooses random value number r_B, and encrypts the ciphertext of Alice's encryption and gets the following two parts: $E_{K_B} = (y_{1B}, y_{2AB})$.

$$y_{1B} = {\alpha_B}^{r_B} \pmod p, y_{2AB} = x{\beta_A}^{r_A}{\beta_B}^{r_B} \pmod p$$

Actually, there is no difference whether Alice or Bob encrypts first; it will get the same ciphertext y_{1A}, y_{1B}, y_{2AB}.

Decryption

If Alice uses her private key to decrypt first,

$D_{K_A}(y_{1A,}, y_{2AB}) = y_{2AB}({y_{1A}}^{k_A})^{-1} = y_{2B} \pmod p$, and then Bob uses his private key to decrypt $D_{K_B}(y_{2B}) = y_{2B}({y_{1B}}^{k_B})^{-1} = x \pmod p$

□ is the original message. Actually there is no difference whether Alice or Bob decrypts first; it could use the following formula to express the whole multi-party decryption.

$$D_{K_A,K_B}(y_{1A,}, y_{1B}, y_{2AB}) = y_{2AB}({y_{1A}}^{k_A})^{-1}({y_{1B}}^{k_B})^{-1} = x \pmod p$$

Application: Card Dealing of Mental Poker game

(1) Alice (a card dealer) encrypts original cards with her secret key one by one. The set of encrypted cards is $\{E_A(1), \square, E_A(52)\}$ and she sends them to Bob.

(2) Bob choose 5 cards at random say, $\{E_A(3), E_A(13), E_A(23), E_A(24), E_A(25)\}$, encrypts them. And he sends the $\{E_{AB}(3), E_{AB}(13), E_{AB}(23), E_{AB}(24), E_{AB}(25)\}$ back to Alice.

(3) Alice decrypts each element of the set, and sends the resulting set, $\{E_B(3), E_B(13), E_B(23), E_B(24), E_B(25)\}$, back to Bob.

(4) Bob decrypts the set to get his hand $\{3,13,23,24,25\}$.

In next section, we apply card dealing method that used [ZV03] scheme to watermark generation protocol.

---

should be able to claim to have any cards but those dealt. Here, the power of commutative cryptosystems is utilized. The advantage of [ZV03] is that there is no information leakage and we can extend to multi-party encryption and decryption system without losing generality because the final ciphertext is the same even if a different order is used for encryption.

4 Proposed Buyer-Seller Watermarking Protocol

In this section, we describe buyer-seller watermarking protocol without trusted third party, which is a modified scheme of [JK02] such that the watermark certification authority issues a buyer's unique watermark upon request and the encrypted watermark with the buyer's anonymous public key is received. Our scheme is based on [MW01] scheme as embedding method and [CK97] scheme as building block for collusion resistance.

4.1 Preliminary

[Preprocessing]

Let $p(\leq n\ bits)$ be a large prime such that $q = (p-1)/2$ is also a prime. Let G be a group of order $p-1$, and let g be a generator of G such that computing discrete logarithms to the base g is difficult.

[Roles of Each entity]

The entities of our scheme consist of the watermark certification authority, seller, buyer, and an arbiter. The role (or notation) of each entity is as follows.

Watermark certification authority
- Carol is short for the watermark certification authority.
- She issues watermark to buyers upon request and certify it.

Arbiter
- He/she should be convinced in trials.
- It should be possible to convince anyone as long as they know a few specific public keys.

Seller
- She (Alice) is the agent selling the contents.

Buyer
- He (Bob) is the buyer that can buy contents anonymously.

All participants (Alice, Bob, and Carol) have a pair of a secret key and a public key $(sk, pk) : [(sk_A, pk_A), (sk_B, pk_B), (sk_C, pk_C)]$ such that $pk = g^{sk} \bmod p$, all of which have been registered with appropriate certificate authority (CA).

[Notations]

We assume that the content being sold is a still image, though in general the protocol is also applicable to audio and video data like [MW01] scheme and [JK02] scheme for ease of exposition. We establish some notation as follows.

- X : Original image to be a vector of "features", $X = \{x_1, \square, x_m\}$.
- W : Watermark as a vector of "watermark elements", $W = \{w_1, \square, w_n\}$.
- X', X'' : Watermarked image

- $X \oplus W = \{x_1 \oplus w_1, \square, x_n \oplus w_n, x_{n+1}, \square, x_m\}$, $m \geq n$
- $\oplus$: Insertion operation
- E_H / D_H : Encryption/decryption algorithm with homomorphic property
- E_T / D_T : Encryption/decryption algorithm with property of commutative cryptosystem

The proposed protocol consists of the following steps: Watermark generation step for generation of a buyer's unique and valid watermark, watermark insertion step for making a watermarked content of buyers, copyright violator identification step in order to identify dishonest buyers. We introduced two cryptosystems such as cryptosystems with homomorphic property and commutative property in order that the watermark certification authority cannot know which watermark the buyer chose and sellers can embed valid watermark into content without disclosing it.

STEP 1. Watermark Generation

1. Bob chooses secret random sk^*_{B1}, sk^*_{B2} in Z_p such that $sk^*_{B1} \cdot sk^*_{B2} = sk_B \in Z_p$. Bob sends pk_B, pk^*_B $(pk^*_B = g^{sk^*_{B1}})$ and sk^*_{B2} $(E_{H\,pk_C}(sk^*_{B2}))$ encrypted by using the Carol's public key pk_C. Bob convinces Carol of zero-knowledge of possession of sk^*_{B1}. The proof given in [Ch87] for showing possession of discrete logarithms may be used here.
2. Carol first decrypts $E_{H\,pk_C}(sk^*_{B2})$ using his private key sk_C and checks that $pk_B^{*\,sk^*_{B2}} = pk_B \pmod p$ with the Bob's public key pk_B certified by CA. If it is verified, then Carol issues $k(\geq 2)$ watermarks $(W_1, W_2, \square, W_k)$ as follows.
   (1) Carol generates valid k watermarks $(W_1, W_2, \square, W_k)$ randomly. Note that $W_i = \{w_{i1}, w_{i2}, \square, w_{in}\}$

   Remark 1: Here, the watermark certification authority issues k watermarks, where Bob would choose one out of k watermarks. The choice of k implies a trade off between correctness and efficiency. In such case, probability that watermark certification authority can know watermark that a buyer chose would be equal to $1/k$. We use a specific construction which introduced a spread-spectrum watermarking techniques[4] proposed by Cox et al [CD98]. Each W_i of this protocol and W of Cox scheme has the same property.

   (2) Carol makes k pair $(P_1, \square, P_k)$ of watermarks and its signature as equation (1). First she encrypts each watermark (W_i) with Bob's anony-

[4] Cox et al., embed a set of independent real numbers $W = \{w_1, \square, w_n\}$ drawn from a zero mean, variance 1, Gaussian distribution into the m largest DCTAC coefficients of an image. Results reported using the largest 1000 AC coefficients show the technique to be remarkably robust against various image processing operations and after printing and rescanning and multiple-document (collusion) attack.

mous public key pk_B^*, along with a digital signature $sign_{sk_C}(E_{H\,pk_B^*}(W_i)), i = \{1,2,\square,k\}$ that certifies the validity of the watermarks and also ensures that pk_B^* was used to encrypt the watermark as a public key. Then she generates a pair of key (r_C, r_C') and encrypts each $P_1, \square, P_k$ with it using encryption scheme E_T. Here r_C is the encryption key and r_C' is a decryption key corresponding with r_C. She sends them $(EP_1, \square, EP_k)$ to Bob.

$$\begin{gathered} ew_1 = E_{H\,pk_B^*}(W_1), \square, ew_k = E_{H\,pk_B^*}(W_k) \\ s_1 = sign_{sk_C}(ew_1 \,\|\, pk_B^*), \square, s_k = sign_{sk_C}(ew_k \,\|\, pk_B^*) \\ P_1 = (ew_1 \,\|\, s_1), \square, P_k = (ew_k \,\|\, s_k) \\ EP_1 = E_{T\,r_C}(P_1), \square, EP_k = E_{T\,r_C}(P_k) \end{gathered} \tag{1}$$

3. Now, Bob cannot know the hidden watermark and its signature $(P_1, \square, P_k)$ because $(EP_1, \square, EP_k)$ are encrypted with Carol's secret key. Bob generates a pair of key (r_B, r_B'). r_B is encryption key and r_B' is a decryption key corresponding with r_B. He chooses one among them, $(EP_1, \square, EP_k)$. Suppose Bob chose $EP_3 = E_{T\,r_C}(P_3) = E_{T\,r_C}(ew_3 \,\|\, s_3)$. He encrypts it with r_B using encryption scheme E_T. He sends $E_{T\,r_B}(EP_3)$ it back to Carol.

$$E_{T\,r_B}(EP_3) = E_{T\,r_B}(E_{T\,r}\ (ew_3 \,\|\, s_3)) \tag{2}$$

4. Carol computes equation (3) and records pk_B, pk_B^* at his table $Table_C$. Then she sends $E_{T\,r_B}(ew_3 \,\|\, s_3)$ back to Bob. She is not able to know which watermark Bob chose because $(P_1, \square, P_k)$ is re-encrypted with Bob's secret key. Here, information to be sent to Bob is encrypted with only Bob's secret key.

$$D_{T\,r_C'}\{E_{T\,r_B}(EP_3)\} = D_{T\,r_C'}\{E_{T\,r_B}(E_{T\,r_C}(P_3)\} = E_{T\,r_B}(P_3) = E_{T\,r_B}(ew_3 \,\|\, s_3) \tag{3}$$

5. Bob decrypts $E_{T\,r_B}(ew_3 \,\|\, s_3)$ with r_B' and verifies s_3, ew_3 with Carol's public key pk_C and his secret key sk_{B1}^*.

STEP 2. Watermark Insertion

This is a two-party protocol between Alice and Bob, which proceeds as follows. From here, we will write simply ew, s rather than ew_i, s_i (in the watermark generation protocol, (ew_3, s_3)) that Bob chose.

1. Bob sends ew, s, pk_B^* to Alice.
2. Alice verifies s in order to be assured that ew is indeed a valid watermark verified by the watermark certification authority. If the verification holds,
3. Alice generates a unique watermark for this transaction V, which she inserts into the image X to get the watermarked image X'. Let X denote the image that Bob wants to purchases from Alice. The purpose of the watermark V is to

enable Alice to identify the specific user an illegal copy has potentially arisen from. She then generates a random permutation σ of degree n which she uses to permute the elements of the encrypted watermark $E_{H\,pk_B^*}(W)$ received from Bob. Alice computes $\sigma(E_{H\,pk_B^*}(W)) = E_{H\,pk_B^*}(\sigma(W))$.

4. Alice inserts the second watermark into the already watermarked image X'. Although the watermarks received from Bob is encrypted with Bob's anonymous public key pk_B^*, Alice can embed this second watermark without decrypting $E_{H\,pk_B^*}(W)$. Inserting a watermark in the encrypted domain is possible as we mention that the public key cryptosystems being used is a privacy homomorphism with respect to $\oplus$, the operation that inserts a watermark in the image. That is, Alice computes as follows

$$\begin{aligned} E_{H\,pk_B^*}(X'') &= E_{H\,pk_B^*}(X') \oplus \sigma(E_{H\,pk_B^*}(W)) \\ &= E_{H\,pk_B^*}(X') \oplus E_{H\,pk_B^*}(\sigma(W)) = E_{H\,pk_B^*}(X \oplus V \oplus \sigma(W)) \end{aligned} \tag{4}$$

5. Alice transmits $E_{H\,pk_B^*}(X'')$ to Bob and stores pk_B^*, V, σ, s, ew in her table $Table_A$. $Table_A$ is a table of records maintained by Alice for image X containing one entry for each copy X that she sells.
6. Bob decrypts $E_{H\,pk_B^*}(X'')$ with his secret key sk_{B1}^*.

$$D_{H\,sk_{B1}^*}(E_{H\,pk_B^*}(X'')) = X'' = X' \oplus \sigma(W) = X \oplus V \oplus \sigma(W) \tag{5}$$

Now Bob has a watermarked copy X'' of X that Alice cannot reproduce because she does not know the corresponding private key sk_{B1}^* and W even if she collude with Carol. Also, since Bob does not know σ, he cannot remove $\sigma(W)$ from X'', neither can he remove V which is also unknown to him.

STEP 3. Copyright Violator Identification

On discovering an unauthorized copy of X, say Y, Alice can determine the buyer from whom this copy has originated by detecting the unique watermark that she inserted for each buyer. This is done by means of a watermark extraction function and depends on the watermarking algorithm.

1. When an illegal copy Y of an original image is found, Alice extracts the unique watermark U in Y.
2. For robust watermarks, by computing correlations of extracted watermark U and every watermark stored in $Table_A$, Alice finds V with the highest correlation and obtains the transaction information involving V from the table. If U cannot be matched to any watermark V of the $Table_A$, then this protocol returns failure. Once this V is located in $Table_A$, she reads the buyer's anonymous public key pk_B^* and σ, s, ew. Alice sends them to an arbiter (judge).

3. The judge first verifies $s = sign_{sk_C}(E_{H\,pk_B^*}(W) \| pk_B^*)$ and asks Carol for real identity of an anonymous buyer. The judge would then ask Bob for his private key sk_{B1}^* which he can compute W and check for the presence of $\sigma(W)$ in Y. Actually, Bob needs not reveal his private key sk_{B1}^* because this is undesirable. He could just reveal W to the judge by decrypting $E_{H\,pk_B^*}(W)$. The judge could then verify W by encrypting it with Bob's anonymous public key and checking if it equals to $E_{H\,pk_B^*}(W)$. After verifying W, the judge can then run the watermark extraction algorithm on Y and check if $\sigma(W)$ is indeed present in Y. If $\sigma(W)$ is found in Y, Bob is found guilty otherwise he is innocent.

5 Features and Security Analysis

We discuss and analyze features and security of the proposed scheme according to the list of requirements (Section 1). We assume that all of the underlying primitives are secure. Security of our scheme relies on that of the underlying watermarking algorithm and cryptosystems.

1. **Anonymity**: We assume that the watermark certification authority does not reveal the buyer's real ID if the buyer is honest. In watermark insertion step, the seller knows ew, s, pk_B^*. Finding pk_B would require knowledge of sk_{B2}^*. However, if the encryption algorithm is secure in watermark insertion step, the only way for the seller to find sk_{B2}^* is to compute $\log_g pk_B^*$. But polynomial algorithm proving discrete logarithm problem does not exist, so attacker does not compute sk_{B2}^*. Thus buyer anonymity is guaranteed.
2. **Unlinkability**: Because our scheme executes one-time watermark generation protocol by using an anonymous key pair whenever the buyer buys a contents. This implies that the buyer's purchases are unlinkable.
3. **Traceability**: Due to the properties of the underlying encryption and digital signature techniques, we can assume that a malicious buyer cannot change or substitute a fingerprint generated by the watermark certification authority. The security of traceability is the same as that of [MW01][JK02]. Sellers should insert a watermark V and $\sigma(W)$ in the right manner for her own interest. If she does not correctly insert V, she would not be able to identify the original buyer of an illegal copy. Further a detecting function in the watermark detection must guarantees that the seller can extract the unique watermark W that belongs to a copyright violator. Besides, the buyer cannot remove $\sigma(W)$ from X'' because he does not know σ. Thus the buyer who has distributed digital contents illegally can be traced in our scheme.
4. **No Framing**: Since, to forge Y with the special watermark W, the seller must know either the buyer's private key sk_{B1}^* or the buyer's unique watermark W. In our proposal, only the buyer knows his secret key sk_{B1}^* and his unique watermark

W if computing discrete logarithm is hard and used encryption algorithm (underlying primitives) is secure. Since we use secure commutative cryptosystems in the watermark generation protocol, even the watermark certification authority does not know which watermark the buyer selected. Thus an honest buyer should not be wrongly identified as a copyright violator, because the others cannot recreate the buyer's copy with specific watermark.

5. **No Repudiation**: The buyer accused of reselling an unauthorized copy cannot claim that the copy was created by the seller or a security breach of the seller's system. Since only the buyer know his secret key sk_{B1}^{*} and his unique watermark W, the others cannot recreate the buyer's copy.
6. **Collusion Tolerance**: Our scheme has used [CK97] as a building block. We assumed that this algorithm is secure. And this algorithm is estimated to be highly resistant to collusion attacks [KT98]. Our protocol is secure only as much as the underlying watermarking techniques are secure and robust.
7. **Security against conspiracy attack**: To success in conspiracy attack, seller and the watermark certification authority must know either the watermark W or the buyer's secret key sk_{B1}^{*}. But in our scheme, no one (except the buyer) knows the buyer's unique watermark W and secret key sk_{B1}^{*}. And our protocol is secure against conspiracy attack because the judge of our scheme does not take part in others step except identification step (The arbiter knows just a specific public key). Thus our scheme does not need any trusted third party because all participators' dishonesty can be controlled.

6 Concluding Remarks

To protect both seller and buyer's rights and buyer's anonymity, [JK02] proposed "an anonymous buyer-seller watermarking protocol". But the problem of this protocol is that sellers can recreate the buyer's copy if he/she colludes with the watermark certification authority and the judge. Thus [JK02] scheme must need the trusted third parties for its security. On the contrary, we propose secure buyer-seller watermarking protocol without trusted third party. For it, we apply secure commutative cryptosystems to watermarking protocol. But, drawbacks of our scheme compared with [JK02] are that it requires high computational complexity and communication pass number in watermark generation step. A further direction of this study will be to diminish computational complexity.

References

[BS95] D.Boneh and J.Shaw, "Collusion-secure Fingerprinting for Digital Data", Crypto'95, LNCS 963, Springer-Verlag, 1995, pp. 452–465.

[CD87] D.Chaum, Ivan Bjerre Damgard and Jeroen van de Graaf., "Multiparty Computation Ensuring Privacy of Each Party's Input and Correctness of the Result", Crypto'87, LNCS 293, Springer-Verlag, 1987, pp. 86–119.

[CD98] J.Camenisch and I.Damgard, "Verifiable encryption and applications to group signatures and signatures sharing", Technical Report RS 98-32, Brics, Department of Computer Science, University of Aarhus, Dec.1998.

[Ch87] D.Chaum, "An improved protocol for demonstrating possession of discrete logarithms and some generalizations", Eurocrypto'87, LNCS 304, Springer-Verlag, 1987, pp.127–141.

[CK97] I.J. Cox, J.Kilian, T.Leighton, and T.Shamnon, "Secure spread spectrum watermarking for image, audio and video", IEEE Transactions on Image Processing, vol.6, no 12, pp.1673–1678, 1997.

[GM82] Goldwasser, S. and Micali, S. "Probabilistic Encryption and How to play Mental Poker Keeping Secret All Partial Information", Proceedings of the 14th STOC, pp.365–377, 1982.

[JK02] Hak-Soo Ju, Hyung-Jeong Kim, Dong-Hoon Lee and Jong-In Lim., "An Anonymous Buyer-Seller Watermarking Protocol with Anonymity Control", ICISC2002, LNCS 2587, Springer-Verlag, 2003, pp. 421–432.

[KT98] Joe Killian, F. Thomson Leighton, Lasely R. Matheson, Talal G. Shannon, Robert E. Tarjan, and Francis Zane, "Resistance of Digital Watermarks to Collusive attacke", 1998.

[Ne83] Neal.R.Wanger, "Fingerprinting", IEEE Symposium on Security and Privacy, 1983.

[MW01] N.Memon and P.W.Wong, "A Buyer-Seller Watermarking Protocol", IEEE Transactions on image processing, vol.10, no. 4, pp. 643–649, April 2001.

[PS00] B.Pfitzman and Ahmad-Reza Sadeghi, "Coin-Based Anonymous Fingerprinting", Eurocrypt'99, LNCS 1592. Springer-Verlag, 2000, pp.150–164.

[PS96] B.Pfitzman and M.Schunter, "Asymmetric Fingerprinting", Eurocrypt'96, LNCS 1070, Springer-Verlag, 1996, pp.84–95.

[PW97] B.Pfitzman and W.Waidner, "Anonymous Fingerprinting", Eurocrypto'97, LNCS 1233, Springer-Verlag, 1997, pp. 88–102.

[QN98] L.Qian and K.Nahrstedit, "Watermarking schemes and protocols for protecting rightfuk ownership and customer's rights", J.Visual Commun. Image Represent, vol. 9, pp.194–210, Sept. 98.

[ZV03] Weiliang Zhao, Vijay Varadharajan and Yi Mu, "A secure Mental Poker Protocol Over the internet", Australasian Information Security Workshop 2003. Conference in Research and Practice in Information Technology, Vol.21, 4.Feb.2003.

Using OCSP to Secure Certificate-Using Transactions in M-commerce

Jose L. Muñoz, Jordi Forné, Oscar Esparza, and Bernabe Miguel Soriano

Technical University of Catalonia (UPC)*
Telematics Engineering Department (ENTEL)
1-3 Jordi Girona, C3 08034 Barcelona (Spain)
Phone. +34934010804 Fax.+34934011058
{jose.munoz, jordi.forne, oscar.esparza, ibernabe, soriano}@entel.upc.es

Abstract. The possibility of making the Internet accessible via mobile telephones has generated an important opportunity for electronic commerce. Nevertheless, some deficiencies deter its mass acceptance in e-commerce applications. In order to speed up the information delivery, the use of brokerage systems constitutes an interesting solution. In this paper we review the problem of certificate validation in m-commerce transactions and we present an architecture where a broker is used as OCSP responder for the certificate validation. A modification over OCSP called $\mathcal{H}$-OCSP is also proposed as a way to reduce the computational load and the bandwidth requirements of OCSP which is specially desirable in the wireless environment. The ASN.1 add-on for $\mathcal{H}$-OCSP that makes it inter-operable with the standard OCSP is defined and the behaviour of $\mathcal{H}$-OCSP compared to standard OCSP is evaluated.

Keywords: broker, m-commerce, certification, certificate status checking, OCSP

1 Introduction

The access to the Internet by means of mobile devices potentially increases the number of users of e-commerce. One of the novelties of m-commerce is the possibility of attracting clients in the neighborhoods of commercial and/or service centers by providing them with appropriate information.

A way to ease and make more efficient the access to information from mobile terminals is to use a broker between the terminal and the wired network. Figure 1 shows a possible broker-based architecture. The broker re-uses the information and sends useful data in a predictive mode to wireless users, reducing data traffic in the wireless link.

* This work has been supported by the Spanish Research Council under the project DISQET (TIC2002-00818).

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 280–292, 2003.

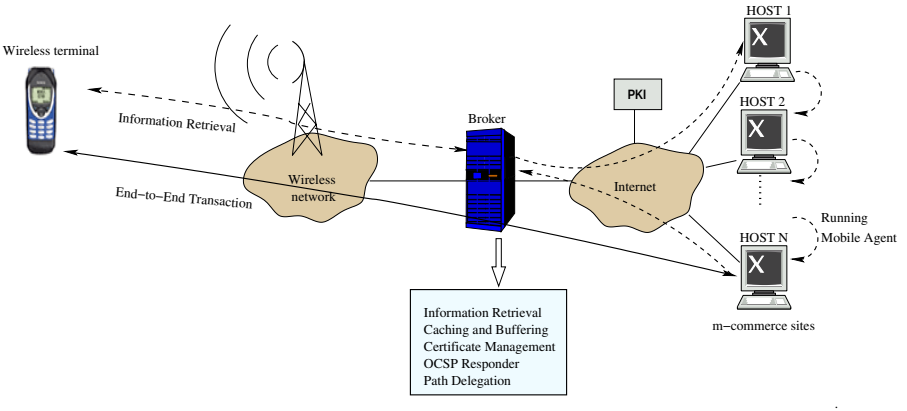

Fig. 1. Broker Architecture for a Wireless Scenario

The main functions of the broker are detailed below:

- *Intermediate storage*: the broker stores a copy of the information that it receives. This will allow its re-use by other users. Therefore, the information must be organized inside proxy and cache systems.
- *Customer Relationship Management infrastructure*: the knowledge of the user profile allows to manage the user's needs for information, and to deliver information in a predictive fashion and dynamic tracking of clusters of data. A user profile could be managed and updated dynamically, processing information from different sources. In wireless environment, CRM (Customer Relationship Management) can profit from the facilities of the push mechanism of mobile communication protocols.
- *Information search and retrieval*: tasks such as searching, advising, contacting, comparing, filtering and facilitating access to databases perfectly fit for mobile agent technology. In this sense, agents can perform some tasks on behalf of a user while the mobile device remains off-line, which is very desirable on a noisy weak link with unpredictable disconnection.
- *Certificate management*: a Public Key Infrastructure (PKI) is required to provide the m-commerce transactions with the security services such as integrity, privacy, non-repudiation, authentication and access control. The broker is appropriate for storing and managing digital certificates, and combined with OCSP (On line Certificate Status Protocol) [8] timely information about the status of certificates can be obtained. The rest of the paper deals with this function.

Digital certificates are usually used to identify parties involved in e-commerce and m-commerce transactions, as well as to provide end-to-end security in these transactions. A certificate is a digital document signed by a Trusted[1] Third

[1] Trust in a principal can be defined as a belief that the principal, when asked to perform an action, will act according to a pre-defined description. In particular, this

Party (TTP) called "issuer". Certificates are tamper-evident, in other words, they can be easily read but they cannot be modified without making the signature invalid. Moreover, they are unforgeable because only the issuer can produce the signature. There are several kinds of certificates but the most widely used are Identity Certificates (ICs), whose main function is to bind a public-key with an identity. An IC states an association between a name called a *Distinguished Name* (DN) and the user's public-key. Therefore, the authentication of the certificate relies on each user possessing a unique DN. DNs use the X.500 standard [11] and are intended to be unique across the Internet. On the other hand, the X.509 standard [5,12] defines what information can go into a certificate and its data format. All X.509 certificates have the following data, in addition to the signature:

- *Version.* This field identifies which version of the X.509 standard applies to this certificate, which affects what information can be specified in it. So far, three versions have been defined.
- *Serial Number.* The entity that created the certificate is responsible for assigning it a serial number to distinguish it from other certificates it issues.
- *Signature Algorithm Identifier.* Identifies the asymmetric algorithm used by the issuer to sign the certificate.
- *Issuer Name.* The DN of the issuer.
- *Validity Period.* Each certificate is valid only for a limited amount of time. It is not valid prior to the activation date and it is not valid beyond the expiration date.
- *Subject Name.* The DN of the entity whose public-key the certificate identifies.
- *Subject Public Key Information.* This is the public-key of the entity being named, together with an algorithm identifier which specifies which public-key crypto system this key belongs to and any associated key parameters.

It has been made also a profile for the use of ICs in the Internet by IETF in [4]. However, before performing a transaction it is crucial that the user makes sure that all certificates involved are valid, in opposite case, the transaction should be rejected since the securing mechanisms rely on the trust of the underlying certificates.

The rest of the paper is organized as follows: in Section 2 we introduce the problem of certificate validation in m-commerce transactions. In Section 3 we propose $\mathcal{H}$-OCSP as a way to reduce the computational load in the broker. In Section 4 we define the ASN.1 add-on for $\mathcal{H}$-OCSP that makes it inter-operable with the standard OCSP. In Section 5 we evaluate the behavior of $\mathcal{H}$-OCSP compared to standard OCSP. Finally, we conclude in Section 6.

---

belief implies the belief that the principal will not attempt to harm the requestor independently of the way it fulfills the request [10].

2 Securing Certificate-Using Transactions

Wireless users ask for information or services in the Internet and their requests arrive to the broker that will obtain this information on behalf of the user. After receiving the information, depending on the required service, the user may perform a transaction with a selected m-commerce site. When a user wishes to establish a transaction with a particular m-commerce site, an end-to-end authenticated and private channel is required in order to transfer sensitive data such as a credit card number. Technologies based on ICs, like HTTPS/TLS [1] are being widely used for establishing this kind of secure channels. However, prior to perform a transaction using a certain IC, the user must be sure that the certificate is valid. It must be stressed that the certificate management is a heavy process and that the clients in our environment are resource-limited. For this reason, clients delegate to the broker the processing of the certificates. Notice that the broker is a TTP and it is in general not resource-limited, therefore it is appropriate for storing and managing certificates.

The validation of a certificate comprises two mechanisms: certificate status checking and certification path validation.

The certificate status checking is needed because ICs have a bounded life-time and it may be necessary to revoke an IC before its life-time ends. An IC may be revoked, according to [3], because of:

- The loss or compromise of the associated private key.
- In response to a change in the owner's access rights.
- A change in the relationship with the issuer.
- As a precaution against cryptanalysis.

The two main standards for certificate status checking are Certificate Revocation Lists and Online Certificate Status Protocol.

The *Certificate Revocation List* (CRL) is the most mature approach, it is part of X.509 since its first version [12] and it has also been profiled for the Internet [4]. A CRL is a digitally signed list of revoked certificates, where for each entry in the list the following information is stored: the certificate serial number, the revocation reason and the revocation date. The CRL header includes information about the version, the CRL serial number, the issuer, the algorithm used to sign, the signature, the issuing date, the expiration date and some optional fields called extensions. The CRL is usually distributed to relying parties through non-trustworthy servers called "repositories".

The *Online Certificate Status Protocol* (OCSP) has been proposed by the PKIX workgroup of the IETF [8]. OCSP enables certificate-using applications to determine the revocation state of an identified certificate. The status of certificates is available online through trustworthy servers called "responders" that sign online each response they produce. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate in question until the responder provides a response. For e-commerce, online technologies are interesting not only because they can provide the status data in real-time but also because of billing issues (requests can be used as the basis for billing).

On the other hand, the Certification path validation is necessary when a relying party wants to validate a certificate of an end entity from a different issuer. A certification path is a chain of certificates where the issuer of the first certificate is a trustworthy entity (for example, the issuer of the relying party), and the subject of the last certificate is the end entity. Then, the user needs a mechanism to build and validate this chain of certificates. The problem of verifying a certification path remains an open topic (see "Simple Certificate Validation Protocol (SCVP)" [6] and "DPV and DPD over OCSP" [9]). This study is only focused on the certificate status checking mechanism.

Previous work by the authors have shown that CRLs are not a good choice for distribution of status data in resource-constrained handsets as it is the wireless link because of their high requirement in bandwidth and cache [7]. Therefore, the use of CRLs should be restricted to distribution of data among intermediate entities (e.g. distribution between the issuer CA and the broker). On the other hand, OCSP seems a good choice because a user can retrieve timely status data with a moderate resources usage. Moreover, if there is a need for transaction-specific authorization information, OCSP can provide it by means of its protocol extensions.

3 $\mathcal{H}$-OCSP: Secure and Efficient Status Checking

3.1 Preliminaries

As mentioned earlier, usually an OCSP responder signs online each response it produces. Responses can contain three times in them:

- `thisUpdate` is the time at which the status being indicated is known to be correct.
- `nextUpdate` is the time at or before which newer information will be available about the status of the certificate.
- `producedAt` is the time at which the OCSP responder signed this response.

The client may also send a nonce in its request. In this case, the responder has to include the nonce in the signature computation, and thereby the request and the response are cryptographically binded. However, a denial of service vulnerability is evident with respect to a flood of queries. The production of a signature significantly affects response generation cycle time, thereby exacerbating the situation. Unsigned error responses open up OCSP to another denial of service attack, where the attacker sends false error responses.

In order to alleviate these denial of service vulnerabilities, the OCSP responders may pre-produce signed responses specifying the status of certificates at a certain time [8]. The time at which the status was known to be correct shall be reflected in the `thisUpdate` field of the response. The time at or before which newer information will be available is reflected in the `nextUpdate` field, while the time at which the response was produced will appear in the `producedAt` field of the response. However, the use of precomputed responses allows replay attacks in which an old (good) response is replayed prior to its expiration date but after

the certificate has been revoked. Deployments of OCSP should carefully evaluate the benefit of precomputed responses against the probability of replay attacks. In this sense, notice that it exists the following trade-off:

> *The online signature consumes much processing time. To reduce the possibility of falling into denial of service, the responder may pre-compute the responses and store them in a cache. But pre-produced responses are susceptible of generating reply attacks. To avoid the replay attacks, the responder needs to generate pre-produced responses within a short period of time which consumes many processing resources and this fact may lead the responder again to denial of service.*

3.2 $\mathcal{H}$-OCSP Basics

The result of the previous discussion is that the responder would benefit from a mechanism to pre-produce responses with low processing resources utilization. Below we outline a mechanism that reaches this target. Furthermore, under certain circumstances, the exposed mechanism has also many important benefits for the wireless clients that use OCSP.

The mechanism that we propose is called $\mathcal{H}$-OCSP and it exploits the fact that a OWHF (One Way Hash Function) is at least 10,000 times faster to compute than a digital signature. When an pre-produced response needs to be updated because its `nextUpdate` has become obsolete, a OWHF is performed to update this response instead of a new signature. Using an OWHF will permit the repository to update the responses more frequently without falling into denial of service.

$\mathcal{H}$-OCSP is based on the Even et al. algorithm [2] and it works as follows: when a response is going to be pre-produced, the responder adds a hash-chain to it. The hash chain permits the repository to update the pre-produced response in successive periods with a scarce resources utilization. The hash chain results from applying $d+1$ times a OWHF h over a secret nonce (1)

$$R \xrightarrow{h} R_d \xrightarrow{h} R_{d-1} \xrightarrow{h} \cdots \xrightarrow{h} R_i \xrightarrow{h} \cdots R_2 \xrightarrow{h} R_1 \xrightarrow{h} R_0 \tag{1}$$

Let us define the parameters involved in the process:

`primaryUpdateValue` (R) is the secret nonce. R is only known by the responder (broker) and it is generated for each new pre-produced response.

`maximumUpdateIndex` (d) is the maximum number of periods that a pre-produced response can be updated.

`baseUpdateValue` (R_0) is the last value of the hash chain and it is included in the signature computation of the pre-produced response. R_0 is computed by applying $(d+1)$ times h over R

$$R_0 = h^{d+1}(R) \tag{2}$$

`currentUpdateValue` (R_i) is computed by applying $(d+1-i)$ times h over R

$$R_i = h^{d+1-i}(R) \tag{3}$$

Where i is the number of periods "Δ" elapsed from the documented one (the documented validity period is the period included in the response). Δ is defined as

$$\Delta = \texttt{nextUpdate} - \texttt{thisUpdate} \tag{4}$$

A relying party can verify the validity of a pre-produced response that it is living beyond its documented life-time, say, at time t, where t is included within the period $[\texttt{nextUpdate} + (i-1)\Delta, \texttt{nextUpdate} + i\Delta]$, by checking the equality of equation (5)

$$R_0 = h^i(R_i) \quad with \ i \leq d \tag{5}$$

It must be stressed that to forge a `currentUpdateValue` with the information provided by a previous update value an attacker needs to find a pre-image of a OWHF which is by definition computationally infeasible.

It is obvious that if a pre-produced response becomes stale because the status of the certificate it refers changes, a new signature must be performed, in other words, $\mathcal{H}$-OCSP is only applicable to responses that need to update their documented life-time. However, notice that the majority of the response updates are due to the fact that the documented life-times become obsolete, hence the $\mathcal{H}$-OCSP will have indeed a great impact over the responder performance.

Notice also that $\mathcal{H}$-OCSP produces the desired behavior of pre-produced responses:

> *For one thing, the responder can use pre-produced responses with a small life-time which reduces the risk of replay attacks. For another, the repository can update its pre-produced responses at low cost which reduces the risk of denial of service.*

Finally, it is worth mentioning that when the responder recovers from a crash or is restarted after being down for some reason, the most secure and easiest way to implement the startup environment is to erase all the pre-produced responses from cache. This is the way we perform the startup environment in our $\mathcal{H}$-OCSP responder. However, other implementations may keep the pre-produced responses but do so at their own peril.

Next, we show how to save bandwidth between the client and the responder using $\mathcal{H}$-OCSP. This feature is very interesting in the wireless environment where one of the biggest constrains is the scarce bandwidth that it is available.

Pre-produced responses can be also cached by clients. Let us assume that a previous $\mathcal{H}$-OCSP response for a certain certificate is stored in the client's cache. Then, if the client needs to check the status of the same certificate later, she can ask the responder for a `currentUpdateValue` instead of downloading a standard OCSP response which has a much bigger size. Moreover, if a client performs the majority of his requests for a small set of certificates while other certificates are more rarely requested, it becomes likely to have cached responses for these frequently asked certificates. In this case, the status checking can be performed only sending the `currentUpdateValue` and the best performance of $\mathcal{H}$-OCSP related to bandwidth utilization can be clearly appreciated (see Section 5).

On the other hand, a client may wish to keep an OCSP response as a consumer protection issue. If a client performs an important or a delicate transaction, it may later arise a conflict about the content of the exchanged messages. Usually, digital signatures performed during the transaction are used to solve these kind of non-repudiation trials, therefore it is necessary to have a proof of the status of the involved certificates in the precise moment that the transaction was performed. Notice that the hash chain permits to store old responses for a certain certificate with a reduced storage capacity. This storage can be performed by the broker or even by the client (for the certificates she considers important).

3.3 Security Discussion

Next, we present an informal discussion (rather than formal proofs or demonstrations) about the main security aspects of $\mathcal{H}$-OCSP.

Remember that OCSP can be used with:

- *Cryptographically binded requests and responses.* If the client wants a response cryptographically binded to her request, the repository must take into account the nonce that goes in the request when computing the signed response.
- *Pre-produced responses.* These responses are not bound to any particular request, so the information that a response contains can be re-used to respond as many requests as desired.

Cryptographically binded requests and responses are open to denial of service attacks because of the cost of computing the signatures. In this case the effectiveness of $\mathcal{H}$-OCSP avoiding this kind of denial of service attack fails for the following reasons:

- This mode of operation is very costly to the responder because it has to produce a different response per each client for the same information and store all these data in its cache.
- If the majority of the clients are honest and they collaborate with $\mathcal{H}$-OCSP by using their cache entries for previously asked data, the $\mathcal{H}$-OCSP responder will be more robust than the standard one. However, active adversaries can claim that they do not have previously asked for data (e.g. omitting their cache entries) and they can ask each time for a new response. In this case, the denial of service protection of $\mathcal{H}$-OCSP fails if attackers flood the responder with their requests ($\mathcal{H}$-OCSP will have the same asymptotic behaviour that OCSP).

As a result of the previous discussion, we discourage the use of $\mathcal{H}$-OCSP with cryptographically binded responses (or at least the administrator must be concerned about the risks of using this mode of operation).

On the other hand, when using pre-produced responses, the $\mathcal{H}$-OCSP responder is protected from denial of service attacks for data contained in its cache during (at the most):

$$\Delta_p = \Delta * d \tag{6}$$

We denote Δ_p as the "maximum protection period". Notice that any new request over data already contained in a cached response will not involve much processing capacity usage during the protected interval because the Even et al. algorithm can be executed almost in real time. Therefore the denial of service attack intended by the active adversary can be avoided.

Even though, the responder may have minor denial of service problems at start-up since at that time it must sign each produced response.

4 ASN.1 Add-on for $\mathcal{H}$-OCSP

In this section we address important implementation issues in order to make $\mathcal{H}$-OCSP inter-operable with the standard OCSP. In order to deploy $\mathcal{H}$-OCSP, we need to slightly modify the OCSP protocol. The additional parameters that we introduce are included in extensions. This provides compatibility because support for any specific extension is optional and unrecognized extensions are silently ignored by either the clients or the responders. The protocol is designed to permit inter-operability among standard OCSP clients and responders and $\mathcal{H}$-OCSP clients and responders with any combination among them. The ASN.1 add-on for $\mathcal{H}$-OCSP is presented in Figure 2.

An $\mathcal{H}$-OCSP client can include an extension in the `singleRequestExtensions` with OID `id-pkix-ocsp-base-update-value` to let the responder know that she understands $\mathcal{H}$-OCSP. If the $\mathcal{H}$-OCSP client has an $\mathcal{H}$-OCSP response for the target certificate in its cache, the extension includes the `baseUpdateValue`. Otherwise, the extension is filled with an array of 0 bytes. Upon receipt of a request, a responder determines if the message is well formed, if the repository is configured to provide the requested service and if the request contains the compulsory information.

If an $\mathcal{H}$-OCSP client requests a standard OCSP responder, the extension is silently ignored and the responder responds with a standard OCSP response. If an $\mathcal{H}$-OCSP responder receives a request, it looks for the correspondent extension. Depending on the request extension, the $\mathcal{H}$-OCSP responder will respond with different types of responses:

type-A is an $\mathcal{H}$-OCSP response that includes the `currentUpdateValue`. It is sent to the client if she understands $\mathcal{H}$-OCSP and if the `baseUpda- teValue` provided in the request extension matches the one currently stored by the responder.

type-B (basic) is a standard OCSP basic response. It is sent either if the client does not understand $\mathcal{H}$-OCSP or if it is the first request for a pre-produced response.

type-C contains a basic response plus a `currentUpdateValue`. The basic response includes also the `maximumUpdateIndex` parameter in one of the `singleExtensions` of the response. It is sent if the client understands $\mathcal{H}$-OCSP but it has not a previous $\mathcal{H}$-OCSP response in cache for the target certificate or if the cached response she has is obsolete (i.e. the `baseUpdateValue` does not match the one in the responder).

```
HOCSP DEFINITIONS EXPLICIT TAGS::=
BEGIN
IMPORTS

--PKIX Certificate Extensions
AlgorithmIdentifier, BasicOCSPResponse, id-pkix-ocsp FROM OCSP

AuthorityInfoAccessSyntax, GeneralName, CRLReason FROM PKIX1Implicit88
{iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit-88(2)}

Name, Extensions,Certificate, -- AlgorithmIdentifier, id-kp,
id-ad-ocsp FROM PKIX1Explicit88 {iso(1) identified-organization(3)
dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
id-pkix1-explicit-88(1)};

--Type of H-OCSP responses
TypeAResponse ::= SEQUENCE { currentUpdateValue OCTET STRING }
TypeCResponse ::= SEQUENCE {
  basicResponse BasicOCSPResponse,
  typeAResponse TypeAResponse }
baseUpdateValue ::= OCTET STRING
maximumUpdateIndex ::= INTEGER

-- Object Identifiers (proposal)
id-pkix-hocsp-type-a OBJECT IDENTIFIER ::={ id-pkix-ocsp 8 }
id-pkix-hocsp-type-c OBJECT IDENTIFIER ::={ id-pkix-ocsp 9 }
id-pkix-hocsp-base-update-value OBJECT IDENTIFIER::={ id-pkix-ocsp 10 }
id-pkix-hocsp-maximum-update-index OBJECT IDENTIFIER ::=
{id-pkix-ocsp 11}
```

Fig. 2. ASN.1 add-on for $\mathcal{H}$-OCSP

5 Efficiency Comparison: OCSP vs. $\mathcal{H}$-OCSP

In this section, the authors compare the performance of standard OCSP versus $\mathcal{H}$-OCSP in terms of the down-link bandwidth consumption (responder-to-clients) and the processing capacity utilization in the responder.

The evaluation results have been obtained using our Java implementations of the standard OCSP and the $\mathcal{H}$-OCSP[2]. We have also checked inter-operability of our clients and responders with:

- Euro PKI (*http://ocsp.europki.org\:8026*).
- Open Validation (*http://ocsp.openvalidation.org\:80*).
- Alacris (*http://ocsptest.alacris.com\:3080/ocsp*).

[2] The software for the client and the server can be downloaded from `http://isg.upc.es/cervantes`

The experimental results have been obtained under the following conditions:

- Random revocations and random expirations are used[3].
- The test is configured with a database dynamism of one revocation and one expiration per hour.
- The clients generate 2 status checking requests per hour following an exponential probability density function.
- The target certificates are randomly chosen.
- There are 10,000 clients.
- Each client has a certificate.
- There is an average of 10% revocation.
- It is assumed that it has a group of 10 frequently asked certificates that take the 50% of the status checking requests.
- Pre-produced responses are used (non-cryptographically binded requests and responses).

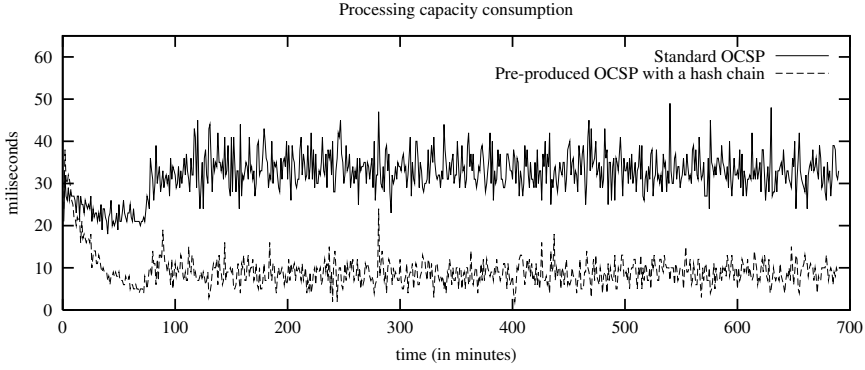

Fig. 3. Processing capacity consumption

Figure 3 shows the computational load of the responder. It can be observed that the computational load at the $\mathcal{H}$-OCSP responder decreases and becomes steady after a few minutes (approx. after 70 min, when the $\mathcal{H}$-OCSP responder starts to take advantage of the pre-computed responses).

In the steady state, the $\mathcal{H}$-OCSP responder consumes around five times less processing capacity than the standard OCSP responder. This is a substantial improvement taking into account that the CPU measurements include not only cryptographic operations but all the operations performed by the java virtual machine (notice that many of these operations such as the garbage collection are very time-consuming).

[3] A revocation implies adding a record to the database of the responder while an expiration implies removing a record from the database (it makes no sense to keep an expired certificate in the revoked certificates database of a responder). Notice

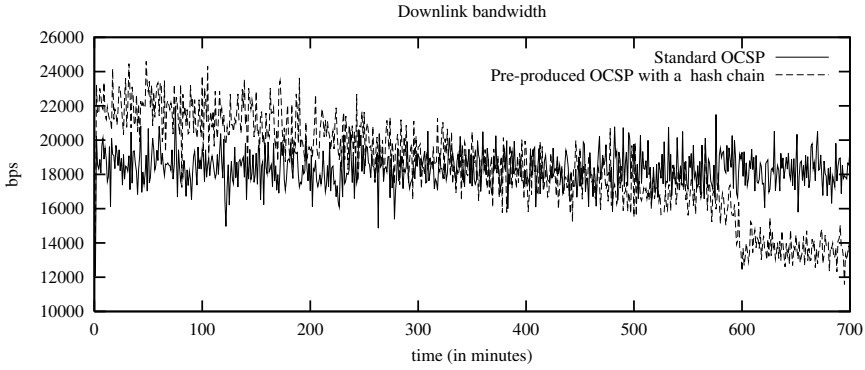

Fig. 4. Down-link bandwidth comparison

Figure 4 shows the measured down-link bandwidth utilization. In the steady state, when the client's cache is fully working, the better performance of $\mathcal{H}$-OCSP can be observed. With less percentage of requests for these frequently asked certificates the performance decreases but in the worse case is approximately as good as in standard OCSP.

6 Conclusions and Future Work

In this paper we introduce the problem of certificate validation in m-commerce transactions. We have shown that many of the validation functions can be delegated to a broker in order to alleviate the resources utilization in the client.

$\mathcal{H}$-OCSP has been proposed as a way to reduce the computational load of the responder. $\mathcal{H}$-OCSP is a hash chain-based mechanism to update pre-produced OCSP responses at a low cost. As a result, an $\mathcal{H}$-OCSP responder is better protected against denial of service attacks than a standard one. Wireless clients can also benefit from $\mathcal{H}$-OCSP by storing the responses of the most used certificates in their cache. However, there is a trade-off between the storing capacity of the wireless terminals and the benefits that $\mathcal{H}$-OCSP can achieve.

We are currently developing efficient cache updating policies for terminals with reduced storing capacity. On the other hand, we have defined the ASN.1 add-on for $\mathcal{H}$-OCSP that makes it inter-operable with the standard OCSP.

Finally, we have evaluated the behavior of $\mathcal{H}$-OCSP compared to standard OCSP and we have shown that $\mathcal{H}$-OCSP requires in general less resources. Although $\mathcal{H}$-OCSP has been designed with the wireless scenario in mind, some of its benefits also apply for the wired internet.

---

that each new record implies pre-producing a new signed response when a status request for this record arrives to the responder.

Acknowledgments. We would like to thank the Spanish Research Council for their support under the Grant TIC2002-00818. We also appreciate the valuable comments and contributions to this article made by the anonymous reviewers.

References

1. T. Dierks and C. Allen. The TLS protocol version 1.0, 1999. RFC 2246.
2. S. Even, O. Goldreich, and S. Micali. Online/offline signatures. *Journal of Criptology*, 9:35–67, 1996.
3. B. Fox and B. LaMacchia. Online Certificate Status Checking in Financial Transactions: The Case for Re-issuance. In *International Conference on Financial Cryptography (FC99)*, number 1648, pages 104–117, February 1999.
4. R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 1999. RFC 2459.
5. ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection – The Directory: Autentication Frameworks, 2000. Technical Corrigendum.
6. A. Malpani, P. Hoffman, P. Housley, and T. Freeman. Simple Certification Validation Protocol (SCVP), December 2002. Internet Draft: draft-ietf-pkix-scvp-11.txt.
7. J.L. Muñoz and J. Forné. Evaluation of Certificate Revocation Policies: OCSP vs. Overissued CRL. In *DEXA Workshops 2002. Workshop on Trust and Privacy in Digital Business (TrustBus02)*, pages 511–515. IEEE Computer Society, September 2002.
8. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, 1999. RFC 2560.
9. Michael Myers. DPV and DPD over OCSP, January 2003. Internet Draft: draft-ietf-pkix-dpvdpd-00.txt.
10. P. Nikander. *An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems.* PhD thesis, Helsinki University of Technology.
11. CCITT Recommendation X.500. The directory overview of concepts, models and services, 1988.
12. ITU/ISO Recommendation X.509. Information technology Open Systems Interconnection – The Directory: Public Key and Attribute Certificate Frameworks, 1997.

Differential Fault Analysis on A.E.S

Pierre Dusart[1], Gilles Letourneux[2], and Olivier Vivolo[2]

[1] LACO (URM CNRS n°6090), Faculté des Sciences & Techniques,
123, avenue Albert THOMAS, 87060 Limoges, France
dusart@unilim.fr,
http://www.unilim.fr/laco
[2] E.D.S.I.
Atalis 1, 1, rue de Paris, 35510 Cesson-Sévigné, France
development@edsi-smartcards.com.fr

Abstract. DFA is no new attack. It was first used by Biham and Shamir who took unfair advantage of DES Feistel structure to carry it out. This structure is not present in AES. Nevertheless, is DFA able to attack AES another way? This article aims at setting out a means of applying DFA to AES that exploits AES internal structure. We can break an AES128 key with ten faulty messages within a few minutes.

1 Introduction

In September 1996, Boneh, Demillo, and Lipton [5] from Bellcore disclosed information about a new type of cryptanalytic attack which exploits computational errors to find cryptographic keys. Their attack is applicable to public key cryptosystems such as RSA, excluding secret key algorithms. In [4], E. Biham & A.Shamir extended this attack to various secret key cryptosystems such as DES, and called it Differential Fault Analysis (DFA). They applied the differential cryptanalysis to Data Encryption Standard (DES) within the frame of hardware fault model.

Since that time 56-bit key has been too short to be secure and worldwide competition between secret key cryptosystems has been raging. The standard which was to replace DES standard had to fulfill the following requirements: be a symmetric cryptosystem with 128 to 256 key sizes, easy to implement with hardware and resilient to linear and differential cryptanalyses. On Oct. 2, 2000, NIST chose Rijndael as Advanced Encryption Standard (AES).

We further assume that the attacker is in possession of a tamperproof-device, so that he can repeat the experiment with the same plaintext and key without applying external physical effects. As a result, he obtains two ciphertexts derived from the same (unknown) plaintext and key, among which one is correct and the other the result of a computation corrupted by a single error occuring during the computation.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 293–306, 2003.

The major criticism of DFA was about its putting into practice possibilities until some authors [3] proved it to be possible. They introduced a fault while the program related to AES was running. A sealed tamperproof device when exposed to certain physical phenomena (e.g., ionizing or microwave radiation) is very likely to cause a fault to happen at a bit in any of the registers at an intermediate stage during the cryptographic computation. In practice, more than one bit can be altered. Whenever the attacker applies DFA attack to the DES, making the most of DES Feistel structure, he knows both differential input and output of the targeted SBox.

When applying DFA to DES, using the Feistel structure of DES, the attacker knows the differential input and output of the target SBox. It is necessary that the attacker should know these differentials to discover a round key byte. With AES, the situation is different because only output differential is known to the attacker. There is no finding immediately the error that alters substitution input. On the other hand, the set of values possibly taken on by the error slipped in can be determined. Knowing that is still not enough. Given that the fault introduced can possibly take 127 values, the round key byte concerned can take as many as 256 values. Thus AES is immune to the classical differential analysis attack. We intend to reduce the set of values possibly taken on by the error introduced assuming that it spreads over at least two distinct bytes used in the SubBytes operation performed through the ciphering process. The error introduced in each SBox input, can possibly take 127 values. As they originate in the same error, only their intersection deserves further consideration. This way the number of possibly committed errors is reduced by half (for a generic case, these sets are different). Either round key byte included in the target Sbox can then take 128 possible values. Key K_{N_r} value can be found by repeating an error at the same state byte. In the end, we proved that AES is vulnerable to differential fault analysis. We implemented the attack on a personal computer. Our analysis program extracted the full AES-128 key by analysing less than 50 ciphertexts.

The document is organized as follows. After briefly descripting AES, we will list a number of DFA-based attack models and then show how to quickly find out the set of values the last round key is likely to take. In the appendix, we illustrate our attack with an example.

The authors thank Joan Daemen for his valuable comments on the article. We are grateful to Cédric Hasard for his help.

2 Brief Description of AES

In this article, we give a description of AES slightly different from [1] as we use a matrix on $GF(2^8)$ to describe a state. Nevertheless, we keep using the notations of [1].

The AES is a block cipher with block length to 128 bits, and support key lengths N_k of 128, 192 or 256 bits. The AES is a key-iterated block cipher : it consists in repeating the application of a round transformation to the state. The number of rounds is represented by N_r and depends on the key length ($Nr = 10$ for 128 bits, $Nr = 12$ for 192 bits and $N_r = 14$ for 256 bits). The AES transforms a state, noted $S \in M_4(GF(2^8))$, (i.e. S is a 4x4 matrix with its coefficients in $GF(2^8)$) into another state in $M_4(GF(2^8))$. The key K is used for generating $N_r + 1$ round keys noted $K_i \in M_4(GF(2^8))$ $(i = 0, 1, \ldots, N_r)$. With AES, a round of an encryption is composed of four main operations: AddRoundKey, MixColumns, SubBytes, ShiftRows.

Remark 1. The representation chosen in [1] of $GF(2^8)$ is $GF(2)[X]/ < m >$, where $< m >$ is the ideal generated by the irreducible polynomial $m \in GF(2)[X]$, $m = x^8 + x^4 + x^3 + x + 1$.

Remark 2. We use three notations, equivalent to one another, to represent an element in $GF(2^8)$:

- $x^7 + x^6 + x^4 + x^2$, the polynomial notation
- $\{11010100\}_b$, the binary notation
- 'D4', the hexadecimal notation

2.1 AddRoundKey for i^{th} Round

The AddRoundKey transformation consists in adding up matrices in $M_4(GF(2^8))$ between the state and the round key of the i^{th} round. We represent by $S_{i,A}$ the state after the i^{th} AddRoundKey.

$$\begin{aligned} M_4(GF(2^8)) &\longrightarrow M_4(GF(2^8)) \\ S &\longmapsto S_{i,A} = S + K_i \end{aligned}$$

2.2 SubBytes for i^{th} Round

The SubBytes transformation consists in applying to each element of the matrix S an elementary transformation s. We represent by $S_{i,Su}$ the state after the i^{th} SubBytes.

$$\begin{array}{c} M_4(GF(2^8)) \longrightarrow M_4(GF(2^8)) \\ S = \begin{pmatrix} S[0] & S[4] & S[8] & S[12] \\ S[1] & S[5] & S[9] & S[13] \\ S[2] & S[6] & S[10] & S[14] \\ S[3] & S[7] & S[11] & S[15] \end{pmatrix} \longmapsto S_{i,Su} = \begin{pmatrix} s(S[0]) & s(S[4]) & s(S[8]) & s(S[12]) \\ s(S[1]) & s(S[5]) & s(S[9]) & s(S[13]) \\ s(S[2]) & s(S[6]) & s(S[10]) & s(S[14]) \\ s(S[3]) & s(S[7]) & s(S[11]) & s(S[15]) \end{pmatrix} \end{array}$$

where s is the non linear application defined by

$$\begin{aligned} GF(2^8) &\longrightarrow GF(2^8) \\ x &\longmapsto s(x) = \begin{cases} a * x^{-1} + b, \text{ if } x \neq 0, \\ b, \text{ if } x = 0. \end{cases} \end{aligned}$$

a is a linear invertible application over $GF(2)$, $a \in M_8(GF(2))$, $*$ is the multiplication of matrices over $GF(2)$ and $x^{-1} = \{b_0 b_1 ... b_7\}_b$ is seen as a $GF(2)$-vector equal to the transposition of the vector $(b_0, \cdots, b_7)$. The value of $b =$ '63'$\in GF(2^8)$ and

$$a = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{pmatrix}.$$

2.3 MixColumns for i^{th} Round

The MixColumns transformation consists in multiplying the state by a fixed matrix A_0 of $M_4(GF(2^8))$. We represent by $S_{i,M}$ the state after the i^{th} MixColumns.

$$\begin{aligned} M_4(GF(2^8)) &\longrightarrow M_4(GF(2^8)) \\ S &\longmapsto S_{i,M} = A_0.S, \end{aligned}$$

where A_0 is defined by

$$A_0 = \begin{pmatrix} \text{'02'} & \text{'03'} & \text{'01'} & \text{'01'} \\ \text{'01'} & \text{'02'} & \text{'03'} & \text{'01'} \\ \text{'01'} & \text{'01'} & \text{'02'} & \text{'03'} \\ \text{'03'} & \text{'01'} & \text{'01'} & \text{'02'} \end{pmatrix}.$$

2.4 ShiftRows for i^{th} Round

The ShiftRows transformation is a byte transposition that cyclically shifts the rows of the state with different offsets. We represent by $S_{i,Sh}$ the state after the i^{th} ShiftRows.

$$\begin{gathered} M_4(GF(2^8)) \longrightarrow M_4(GF(2^8)) \\ S = \begin{pmatrix} S[0] & S[4] & S[8] & S[12] \\ S[1] & S[5] & S[9] & S[13] \\ S[2] & S[6] & S[10] & S[14] \\ S[3] & S[7] & S[11] & S[15] \end{pmatrix} \longmapsto S_{i,Sh} = \begin{pmatrix} S[0] & S[4] & S[8] & S[12] \\ S[5] & S[9] & S[13] & S[1] \\ S[10] & S[14] & S[2] & S[6] \\ S[15] & S[3] & S[7] & S[11] \end{pmatrix}. \end{gathered}$$

3 Attacks on Computation of AES

We describe a list of possible DFA attacks on AES. The attacker is able to introduce a fault into the AES computation process and find out the cryptographic operation output. In all of those attack patterns, one fault means any error possibly several bits long, standing at a byte of the state The goal of these attacks is to find out the key K_{Nr} (and K_{Nr-1} in the case of AES 192 and 256 bits) and hence the key K ([1] for interested readers).

3.1 Models of Attack

All these attacks are based on the basic attack pattern.

Basic attack after the $N_r - 2^{th}$ MixColumns and before the $N_r - 1^{th}$ MixColumns. We introduce a random fault into a definite byte in the state known to the attacker between $N_r - 2^{th}$ MixColumns and the $N_r - 1^{th}$ MixColumns. The fault introduced into the state hits four bytes through last MixColumns. Through SubBytes operation, these four bytes interact with four bytes in ${N_r}^{th}$ round key and result in four differential faults ε'. From each of the four differential faults, we define the set $S_{c,\varepsilon'}$ of values possibly taken on by the initial fault. As the four faults originated in the same initial fault, its real value belongs to the set made up by the intersection of the four previous sets. According to proposition 5, 63 elements at the most constitute that intersection. It stems from proposition 6 that each of the 4 bytes of key K_{Nr} may possibly take 128 values. By iterating the introduction of a fault, the set of values possibly taken on by the keys is reduced by half. After having introduced five faults running, we discover four bytes of the round key. Applying this technique to another row or column, we can manage to discover for other bytes of the last round key. In using about 20 pairs (distorted ciphered output and correct ciphered output), we extract the AES key in full.

Main attack after the $N_r - 2^{th}$ MixColumns and before the $N_r - 1^{th}$ MixColumns. We introduce a random fault into the state at a point unknown to the attacker. That is a generalization of the previous attack. Assuming that the fault may occur at four different places allows us to find out the key K_{Nr}. We thus put ourselves under the conditions of the previous attack so as to determine four sets of Key K_{Nr} possible values. It is necessary that the attack should be repeated using from 40 to 50 distinct pairs (distorted ciphered output and correct ciphered output) for the complete key to be extracted.

Attack after the $N_r - 3^{th}$ MixColumns and before the $N_r - 2^{th}$ MixColumns. We introduce a random fault into the state at a point unknown to the attacker. $N_r - 2^{th}$ ShiftRows spreads each of the four faults over another column of the state. The last MixColumns propagates each fault contained in a column to the whole of it. We can apply the result of the basic attack to every fault contained in a column in order or determine the sets of values possibly taken on by every byte of the last round key. In such a case ten faults are required to get key K_{Nr}.

Attack on hardware device. It is possible to apply the previous patterns of attack to hardware device. Suppose that you can physically modify a hardware AES device. Firstly, compute the outputs from around ten random plaintexts with an AES device. Secondly, modify for instance the component design by

cutting wires lying between two bytes and grounding them (or Vcc) temporarily two rounds before the process ends. It amounts to having a byte of round $N_r - 2$ with a '00' (or 'FF') value. Compute another time the same plaintexts as previously with the tampered device. When the input is a random plaintext, the error generated is a random one. Proceeding along as set out in the previous paragraph, we can extract key K_{Nr}.

3.2 Basic Principle of Attacks

Let us analyse basic attack, we denote by F the erroneous state. Now we are going to describe each step of the state from the $N_r - 1^{th}$ MixColumns to the end. Assume that we replace the first element of the state by an unknown value. Let $\varepsilon \in GF(2^8) - \{0\}$, and get

$$F_{N_r-1,Sh}[0] = S_{N_r-1,Sh}[0] + \varepsilon.$$

$$F_{N_r-1,Sh} = S_{N_r-1,Sh} + \begin{pmatrix} \varepsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

$$F_{N_r-1,M} = S_{N_r-1,M} + A_0.\begin{pmatrix} \varepsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = S_{N_r-1,M} + \begin{pmatrix} \text{'02'}.\varepsilon & 0 & 0 & 0 \\ \varepsilon & 0 & 0 & 0 \\ \varepsilon & 0 & 0 & 0 \\ \text{'03'}.\varepsilon & 0 & 0 & 0 \end{pmatrix}.$$

$$F_{N_r-1,A} = S_{N_r-1,A} + \begin{pmatrix} \text{'02'}.\varepsilon & 0 & 0 & 0 \\ \varepsilon & 0 & 0 & 0 \\ \varepsilon & 0 & 0 & 0 \\ \text{'03'}.\varepsilon & 0 & 0 & 0 \end{pmatrix}.$$

We can define $\varepsilon_0^{'}$, $\varepsilon_1^{'}$, $\varepsilon_2^{'}$, $\varepsilon_3^{'}$ (the differential faults) by the equations

$$\begin{cases} s(x_0 + \text{'02'}.\varepsilon) = s(x_0) + \varepsilon_0' \\ s(x_1 + \varepsilon) = s(x_1) + \varepsilon_1' \\ s(x_2 + \varepsilon) = s(x_2) + \varepsilon_2' \\ s(x_3 + \text{'03'}.\varepsilon) = s(x_3) + \varepsilon_3' \end{cases} \tag{1}$$

Consequently

$$F_{N_r,Su} = S_{N_r,Su} + \begin{pmatrix} \varepsilon_0^{'} & 0 & 0 & 0 \\ \varepsilon_1^{'} & 0 & 0 & 0 \\ \varepsilon_2^{'} & 0 & 0 & 0 \\ \varepsilon_3^{'} & 0 & 0 & 0 \end{pmatrix}.$$

$$F_{N_r,Sh} = S_{N_r,Sh} + \begin{pmatrix} \varepsilon_0^{'} & 0 & 0 & 0 \\ 0 & 0 & 0 & \varepsilon_1^{'} \\ 0 & 0 & \varepsilon_2^{'} & 0 \\ 0 & \varepsilon_3^{'} & 0 & 0 \end{pmatrix}.$$

$$F_{N_r,A} = S_{N_r,A} + \begin{pmatrix} \varepsilon_0^{'} & 0 & 0 & 0 \\ 0 & 0 & 0 & \varepsilon_1^{'} \\ 0 & 0 & \varepsilon_2^{'} & 0 \\ 0 & \varepsilon_3^{'} & 0 & 0 \end{pmatrix}.$$

$F_{N_r,A}$ is the erroneous output of a cipher. Comparing the states $F_{N_r,A}$ with $S_{N_r,A}$, the values of $\varepsilon_0^{'}$, $\varepsilon_1^{'}$, $\varepsilon_2^{'}$ and $\varepsilon_3^{'}$ can be easily found.

The only operation that could give a clue about the key K_{N_r} is the last SubBytes transformation. Consequently we have four equations where $x_0, x_1, x_2, x_3, \varepsilon$ are unknown variables. We want to solve the system of equations (in x_i and ε) (1). All these equations belong to a generalized equation :

$$s(x + c.\varepsilon) + s(x) = \varepsilon', \tag{2}$$

where $c =$'01', '02' or '03'. Let us analyse it.

Remark 3. The map defined by $x \mapsto x^{-1}$ in $GF(2^8)$ is differentially 2 or 4 uniform [7]. This map has other favorable cryptographic properties: large distance from affine functions, high non-linear order and efficient computability.

Definition 1. *Consider the linear application in $GF(2)$:*

$$\begin{aligned} l : GF(2^8) &\longrightarrow GF(2^8) \\ x &\longmapsto x^2 + x \end{aligned}$$

Let us represent by $E_1 = Im(l)$ the $GF(2)$-vector space image of l. We have $dim_{GF(2)}(E_1) = 7$. If $\theta \in E_1$, then there are two solutions $x_1, x_2 \in GF(2^8)$ to the equation $x^2 + x = \theta$, and the solutions satisfy the equation $x_2 = x_1 + 1$.

Definition 2. *Let $\lambda \in GF(2^8)$, $\lambda \neq 0$ and define ϕ_λ a $GF(2)$-vector spaces isomorphism:*

$$\begin{aligned} \phi_\lambda : GF(2^8) &\longrightarrow GF(2^8) \\ x &\longmapsto \lambda.x \end{aligned}$$

and let $E_\lambda = Im(\phi_\lambda|_{E_1})$ be the $GF(2)$-vector space image of ϕ_λ restricted to E_1. Moreover $dim_{GF(2)}(E_\lambda) = 7$.

Proposition 1. *There is a bijective application ϕ between $E_1^*(= E_1 - \{0\})$ and $S_{c,\varepsilon'}$.*

$$\begin{aligned} \phi : E_1^* &\longrightarrow S_{c,\varepsilon'} \\ t &\longmapsto (c(a^{-1} * \varepsilon').t)^{-1}. \end{aligned}$$

$S_{c,\varepsilon'}$ have 127 elements.

Proof. Let $\varepsilon \in S_{c,\varepsilon'}$, then $\exists x \in GF(2^8)$ such that (2) holds.
Let us assume $x \neq 0$ and $x \neq c.\varepsilon$, we get

$$x^2 + c.\varepsilon.x = (a^{-1} * \varepsilon')^{-1}.c.\varepsilon.$$

We represent by $t = x.(c.\varepsilon)^{-1} \in GF(2^8) - \{0\}$, then we have

$$t^2 + t = (a^{-1} * \varepsilon')^{-1}.(c.\varepsilon)^{-1}. \tag{3}$$

Therefore $(a^{-1} * \varepsilon')^{-1}(c.\varepsilon)^{-1} \in E_1^*$. Reciprocally for $\theta \in E_1^*$ we can define $(a^{-1} * \varepsilon')^{-1}.(c.\theta)^{-1} \in S_{c,\varepsilon'}$.
Let us assume $x = 0$ or $x = c.\varepsilon$, (2) becomes $a * (c.\varepsilon)^{-1} = \varepsilon'$. We obtain $\varepsilon = ((a^{-1} * \varepsilon').c)^{-1}$. This case is included in the previous one because $1 \in E_1^*$. We observe that when $\theta = 1$, four solutions in x to the equation (2) can be found. In brief, a bijection map exists between E_1^* and $S_{c,\varepsilon'}$:

$$\begin{array}{ccccc} E_1^* & \xrightarrow{\phi_\lambda} & E_\lambda - \{0\} & \longrightarrow & S_{c,\varepsilon'} \\ t & \longmapsto & \lambda.t & \longmapsto & (\lambda.t)^{-1}. \end{array}$$

where $\lambda = c(a^{-1} * \varepsilon')$.

Proposition 2. *The following statements hold for* $\lambda_1, \lambda_2 \in GF(2^8) - \{0\}$*:*

$$dim_{GF(2)}(E_{\lambda_1} \cap E_{\lambda_2}) = \begin{cases} 7 & \text{If } \lambda_1 = \lambda_2 \\ 6 & \text{Otherwise} \end{cases}$$

Proof. Proving that following lemma 1 holds true is enough to prove Proposition 2 holds true too.

Lemma 1. *For* $\lambda_1, \lambda_2 \in GF(2^8) - \{0\}$*, we get*

$$E_{\lambda_1} = E_{\lambda_2} \Longleftrightarrow \lambda_1 = \lambda_2.$$

Proof. This lemma is equivalent to this proposition: for $\lambda \in GF(2^8) - \{0\}$,

$$E_\lambda = E_1 \Longleftrightarrow \lambda = 1.$$

Let us prove this statement and assume that $\lambda E_1 = E_1$. Remark that $E_1 = \{t = \{t_7 t_6 \cdots t_0\}_b \in GF(2^8) - \{0\} \ : \ t_7 = t_5\}$. Hence $\{1, x, x^2, x^3, x^4, x^6, x^5 + x^7\}$ is a basis of E_1. Let us multiply the basis vectors v_i of E_1 by $\lambda = \{\lambda_7 \cdots \lambda_0\}_b$. As $\lambda v_i \in E_1$, we have $(\lambda v_i)_7 = (\lambda v_i)_5$. We obtain 7 relations ($\lambda_7 = \lambda_5$, $\lambda_6 = \lambda_4$, $\lambda_5 = \lambda_3 + \lambda_7$, $\lambda_4 = \lambda_6 + \lambda_2 + \lambda_7$, $\lambda_7 + \lambda_3 = \lambda_5 + \lambda_1 + \lambda_6$, $\lambda_5 + \lambda_1 = \lambda_3 + \lambda_4$, $\lambda_6 + \lambda_5 = \lambda_7 + \lambda_3$). We solve this system to obtain $\lambda_7 = \lambda_6 = \lambda_5 = \lambda_4 = \lambda_3 = \lambda_2 = \lambda_1 = 0$. The solution $\lambda = 0$ is not right. We can infer that $\lambda = 1$.

Proposition 3. *For* $\lambda_1, \lambda_2, \lambda_3 \in GF(2^8) - \{0\}$*, we get:*

$$dim_{GF(2)}(E_{\lambda_1} \cap E_{\lambda_2} \cap E_{\lambda_3}) = \begin{cases} 7 & \text{If } \lambda_1 = \lambda_2 = \lambda_3 \\ 6 & \text{If } rank_{GF(2)}\{\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}\} = 2 \\ 5 & \text{Otherwise} \end{cases}$$

Proof. It follows from proposition 2 and this following lemma

Lemma 2. *For $\lambda_1, \lambda_2, \lambda_3 \in GF(2^8) - \{0\}$, we get*

$$E_{\lambda_1} \cap E_{\lambda_3} = E_{\lambda_2} \cap E_{\lambda_3} \Longleftrightarrow \lambda_3^{-1} = \lambda_1^{-1} + \lambda_2^{-1} \text{ or } \lambda_1 = \lambda_2.$$

Proof. 1. $\Leftarrow$

Let $x \in E_{\lambda_1} \cap E_{\lambda_3}$, then $\exists y, t \in E_1$ such that $x = \lambda_1.y = \lambda_3.t$.

$$y = \lambda_1^{-1}.\lambda_3.t = \lambda_2^{-1}.\lambda_3.t + t,$$

$$y - t = \lambda_2^{-1}.\lambda_3.t \in E_1,$$

and

$$x = \lambda_3.t = \lambda_2.(y - t) \in E_{\lambda_2}$$

2. $\Rightarrow$

Let us assume that $\lambda_1 \neq \lambda_2$, and show that $\forall t \in E_1,\ \lambda_3.(\lambda_1^{-1} + \lambda_2^{-1}).t \in E_1$. Let $x = \lambda_3.t \in E_{\lambda_3}$:

- If $x \in E_{\lambda_1}$ then $x \in E_{\lambda_2}$ and $\exists s_1, s_2 \in E_1$ so that $x = \lambda_1.s_1 = \lambda_2.s_2$ and we get $\lambda_3.(\lambda_1^{-1} + \lambda_2^{-1}).t = s_1 + s_2 \in E_1$.
- If $x \notin E_{\lambda_1}$ then $x \notin E_{\lambda_2}$ and we get $\lambda_1^{-1}.x \notin E_1$ and $\lambda_2^{-1}.x \notin E_1$. We have $\lambda_3.(\lambda_1^{-1} + \lambda_2^{-1}).t = \lambda_1^{-1}.x + \lambda_2^{-1}.x \in E_1$ (because $\forall u \notin E_1$ and $\forall v \notin E_1$ then $u + v \in E_1$).

We showed that $E_{\lambda_3.(\lambda_1^{-1}+\lambda_2^{-1})} = E_1$ and with the lemma 1 we get $\lambda_3^{-1} = \lambda_1^{-1} + \lambda_2^{-1}$.

Proposition 4. *Finally for $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in GF(2^8) - \{0\}$, we get:*

$$dim_{GF(2)}(E_{\lambda_1} \cap E_{\lambda_2} \cap E_{\lambda_3} \cap E_{\lambda_4}) = \begin{cases} 7 & \text{If } \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 \\ 6 & \text{If } rank_{GF(2)}\{\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1}\} = 2 \\ 5 & \text{If } rank_{GF(2)}\{\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1}\} = 3 \\ 4 & \text{Otherwise} \end{cases}$$

Definition 3. *We define the set of solutions to (2) in ε by*

$$S_{c,\varepsilon'} = \left\{\varepsilon \in GF(2^8) :\ \exists x \in GF(2^8),\ s(x + c.\varepsilon) + s(x) = \varepsilon'\right\}.$$

Definition 4. *We considered four equations in a different way, but the fault introduced is common to these four equations. This is the reason why we introduce the set of possibly introduced faults S:*

$$\Pi = S_{2,\varepsilon'_0} \bigcap S_{1,\varepsilon'_1} \bigcap S_{1,\varepsilon'_2} \bigcap S_{3,\varepsilon'_3}.$$

Π has a smaller cardinal than $S_{c,\varepsilon}$. This allows one to specify more accurately the set of values possibly taken on the faults. Thus the key can be found out by introducing fewer faults.

Proposition 5. *If two of the four following values* $2^{-1}.\varepsilon'_0,\ \varepsilon'_1,\ \varepsilon'_2,\ 3^{-1}.\varepsilon'_3$ *are not equal, we get*

$$Card\ \left(S_{2,\varepsilon'_0} \bigcap S_{1,\varepsilon'_1} \bigcap S_{1,\varepsilon'_2} \bigcap S_{3,\varepsilon'_3}\right) \leq 63.$$

Proposition 6. *For a differential fault* ε'*, let* $\varepsilon \in \Pi \cap S_{c,\varepsilon'}$ *be a fault value,* $\theta = ((a^{-1} * \varepsilon').c.\varepsilon)^{-1} \in E_1^*$ *and* α, β *the two solutions (in* $GF(2^8)$*) to the equation* $t^2 + t = \theta$*. The possible values of key* $K_{N_r}[i]$ *(for a certain* i*, being the index of element in the state) are*

- *If* $\theta \neq 1$ *then* $K_{N_r}[i]$ *can possibly take on two values*

$$K_{N_r}[i] = s(c.\varepsilon.\alpha) + F_{N_r,A}[i] \text{ or } K_{N_r}[i] = s(c.\varepsilon.\beta) + F_{N_r,A}[i]$$

- *If* $\theta = 1$ *then* $K_{N_r}[i]$ *can possibly take on four values*

$$K_{N_r}[i] = s(c.\varepsilon.\alpha) + F_{N_r,A}[i] \text{ or } K_{N_r}[i] = s(c.\varepsilon.\beta) + F_{N_r,A}[i]$$

$$\text{or } K_{N_r}[i] = b + F_{N_r,A}[i] \text{ or } K_{N_r}[i] = s(c.\varepsilon) + F_{N_r,A}[i]$$

Proof.
- If $\theta \neq 1$ then we know that $\theta \in E_1$, and there are two solutions α, β to $t^2 + t = \theta$. We can deduce two solutions from (2) noted $\{x_1, x_2\}$, where $x_1 = c.\varepsilon.\alpha$ and $x_2 = c.\varepsilon.\beta$.
- If $\theta = 1$, we know that $1 \in E_1$, and there are two solutions α, β to $t^2 + t = 1$. We can deduce two solutions from (2) noted $\{x_1, x_2\}$, where $x_1 = c.\varepsilon.\alpha$ and $x_2 = c.\varepsilon.\beta$. Moreover there are also two trivial solutions to (2): $x_3 = 0$ and $x_4 = c.\varepsilon$.

Once we get a solution x to (2), $K_{N_r}[i]$ value can be easily inferred.

By applying this proposition to the four erroneous elements of the state, we can deduce four sets of values that $K_{N_r}[0]$, $K_{N_r}[7]$, $K_{N_r}[10]$ and $K_{N_r}[13]$ can taken on. By introducing repeatedly a fault into a computation, and considering the intersection of those four sets we soon get the true value for $K_{N_r}[0]$, $K_{N_r}[7]$, $K_{N_r}[10]$ and $K_{N_r}[13]$.

3.3 Probability Complexity

We want to know how many pairs we need to crack the cipher.

Proposition 7. *In average, 9 pairs are required to find 4 bytes of the* K_{N_r} *round key. Alike, 11 pairs are required for the Basic attack, 9 for the Extended ones and 34 for the main one.*

Proof. Denote by Card K the cardinal of possible values taken on by any byte of K_{N_r}. Under propositions 4 and 6, supposing they have been distributed at random, probabilities are as follows:

- $\frac{256 \cdot 255 \cdot 254 \cdot 253}{256^4}$ that Card $K = 32$
- $\frac{C_4^2 \cdot 256 \cdot 255 \cdot 254}{256^4}$ that Card $K = 64$

– $\frac{(C_4^3+C_4^2/2)\cdot 256\cdot 255}{256^4}$ that Card $K = 128$
– $\frac{256}{256^4}$ that Card $K = 256$

On average, Card $K = 32.75146103$ when the position of the fault is known. In general, such information is not available so the four possibilities have to be tried: Card $K = 4 \cdot 32.75146103 = 131.0058441$. Each time, the set of possible values is divided by $\frac{256}{131.0058441} \approx 1.954111298$. To bring the number of possibilities down to one, $\ln 256 / \ln 1.954111298 \approx 8.277180940$ pairs are required. Hence, nine faulty ciphertexts have to be used to find 4 bytes of the K_{N_r} round key. In the Extended attacks, 16/4=4 more pairs are required but the four errors are treated simultaneously. Hence in these cases, nine faulty ciphertexts have to be used to find the whole K_{N_r} round key. In the basic attack, the errors have to be distribued all over the other bytes; 16/4=4 more pairs are required:

$$4\frac{\ln 256}{\ln \frac{256}{32.75146103}} \approx 10.78707691.$$

A Example

We will be using the same example as in Appendix B of [1]. The following diagram shows the values in the final states for a block length and a Cipher Key length of 16 bytes each (i.e., $Nb = 4$ and $Nk = 4$).

`Input=` '32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34'
`Cipher Key=` '2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C'
`Output=` '39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32'

The spreading of the fault is highlighted:

After ShiftRows 9				Fault injected 1E				After MixColumns					K_9			
87	F2	4D	97	99	F2	4D	97	7B	40	A3	4C	⊕	AC	19	28	57
6E	4C	90	EC	6E	4C	90	EC	29	D4	70	9F		77	FA	D1	5C
46	E7	4A	C3	46	E7	4A	C3	8A	E4	3A	42		66	DC	29	00
A6	8C	D8	95	A6	8C	D8	95	CF	A5	A6	BC		F3	21	41	6E

After AddRoundKey 9				After SubBytes 10				After ShiftRows 10					value of K_{10}			
D7	59	8B	1B	0E	CB	3D	AF	0E	CB	3D	AF	⊕	D0	C9	E1	B6
5E	2E	A1	C3	58	31	32	2E	31	32	2E	58		14	EE	3F	63
EC	38	13	42	CE	07	7D	2C	7D	2C	CE	07		F9	25	0C	0C
3C	84	E7	D2	EB	5F	94	B5	B5	EB	5F	94		A8	89	C8	A6

Output with Faults			
DE	02	DC	19
25	DC	11	3B
84	09	C2	0B
1D	62	97	32

The error injected into the state, generates four further errors (differential faults) in the final state.

Output with faults					Output without fault					Error			
DE	02	DC	19		39	02	DC	19		E7	00	00	00
25	DC	11	3B	⊕	25	DC	11	6A	=	00	00	00	51
84	09	C2	0B		84	09	85	0B		00	00	47	00
1D	62	97	32		1D	FB	97	32		00	99	00	00

The differential faults are $\varepsilon'_0 =$ 'E7', $\varepsilon'_1 =$ '51', $\varepsilon'_2 =$ '47' and $\varepsilon'_3 =$ '99'. The following four equations have now to be worked out:

$$\begin{aligned} s(x_0 \oplus \text{'02'}.\varepsilon) &= s(x_0) \oplus \text{'E7'} \\ s(x_1 \oplus \varepsilon) &= s(x_1) \oplus \text{'51'} \\ s(x_2 \oplus \varepsilon) &= s(x_2) \oplus \text{'47'} \\ s(x_3 \oplus \text{'03'}.\varepsilon) &= s(x_3) \oplus \text{'99'} \end{aligned}$$

As defined previously,

$$E_1^* = \{\text{'01'..'1F','40'..'5F','A0'..'BF','E0'..'FF'}\}.$$

Let

$$\begin{aligned} \lambda_0 &= \text{'02'}.(a^{-1} * \text{'E7'}) = \text{'12'} \\ \lambda_1 &= \text{'01'}.(a^{-1} * \text{'51'}) = \text{'7C'} \\ \lambda_2 &= \text{'01'}.(a^{-1} * \text{'47'}) = \text{'65'} \\ \lambda_3 &= \text{'03'}.(a^{-1} * \text{'99'}) = \text{'B0'} \end{aligned}$$

We have a single linear relation over $GF(2)$ between $\lambda_0, \lambda_1, \lambda_2, \lambda_3$: $\lambda_0^{-1} \oplus \lambda_3^{-1} = \lambda_2^{-1}$. Therefore we get

$$\text{card}\left(S_{2,\text{'E7'}} \bigcap S_{1,\text{'51'}} \bigcap S_{1,\text{'47'}} \bigcap S_{3,\text{'99'}}\right) = 2^5 - 1 = 31.$$

Using the relation $S_{c,\varepsilon'} = \{(c.(a^{-1} * \varepsilon').t)^{-1}, \quad t \in E_1^*\}$, we can easily (and quickly!) compute

$$\begin{aligned} &S_{2,\text{'E7'}} \bigcap S_{1,\text{'51'}} \bigcap S_{1,\text{'47'}} \bigcap S_{3,\text{'99'}} \\ &= \{\text{'01', '04', '13', } \textbf{'1E'}\text{, '21', '27', '33', '3B', '48', '4D', '50', '53', '55', '5D',} \\ &\quad \text{'64', '65','7E', '7F', '80', '83', '8D', '8F', '93', 'A7', 'A8', 'A9', 'AB',} \\ &\quad \text{'B3', 'B8', 'C9', 'F6'}\} \end{aligned}$$

Using the proposition 6, we get a set of values possibly taken on by $K_{10}[0]$ (the true value is 'D0'):

$K_{10}[0] \in$ {'03', '06', '09', '0C', '10', '15', '1A', '1F', '21', '24', '2B', '2E', '32', '37', '38', '3D', '43', '46', '49', '4C', '50', '55', '5F', '61', '64', '6B', '6E', '72', '77', '78', '7D', '83', '86', '89', '8C', '90', '95', '9A', '9F', 'A1', 'A4', 'AB', 'AE', 'B2', 'B7', 'B8', 'C3', 'C6', 'C9', 'CC', **'D0'**, 'D5', 'DA', 'DF', 'E1', 'E4', 'EB', 'EE', 'F2', 'F7', 'F8', 'FD'}

By introducing a second fault into the very same place in the state, we reduce by half the set of values possibly taken on by $K_{10}[0]$. Introducing a fault five times over, we can find out the one and only true value of $K_{10}[0]$. Of course, we can also analyse the other three bytes $K_{10}[7]$, $K_{10}[10]$ and $K_{10}[13]$ as we analyse the first one. Doing so, we can find out the true values of 4 bytes in key K_{10}. In order to determine the other 4 bytes of the key K_{10}, we have to introduce a fault into any other place in the state and repeat the above described process.

B Example 2

We compute ten pairs of (correct/faulty) ciphertexts with an AES-128 program. We know that we injected a fault into a byte between the MixColumn7 and MixColumn8 operations. Still we know neither its position nor its value (for the readers interesting in repeating those computations, the fault injected replaces State[0] by 'FF' just before MixColumn8 in the following examples).

Correct CipherText	Faulty CipherText
'467A7363D54E58BB25B135FABFA0EA49'	'4F41429299FFBE374514034F07BF4B19'
'9EEE064F55D3B0F5DDC0002E33CDCBEE'	'DF0C7EBA22B9131D83ADE91D223ADD6F'
'5EB4F21A7493ED8EA431B8E6B73FA924'	'2A2B37C7B08482E43063040D357E7F92'
'1A6FC7471E2A43460AE4F29296CCB731'	'A83C77CE284BCAF64DDE12DF58D8B9DB'
'77111043CE69C252E7219FBB12371CD66'	'B7FF53C4D24FF23DF8618B229F8522CB'
'3253954160E455152D77F8A0748B0CEB'	'CA499E9FB8BC82E3120C489FACDC654D'
'538FFA5AD396AE973EDB8C50B44EC54C'	'5C655A7BDE74DED49BE0D36BF27662B8'
'16633326264442DA55F3362384FF1144B'	'51116D1D351518FC7021931A20AC49A0'
'F9CC9D6B31BC0EA27D4E239DBBC943CD'	'75EC4D4F1122E1B7F3F8AD578AA2CD11'
'8C7D0ABC6CDD13D0BD268469ED34FADB'	'672A2B22556974C304C8C7DCD499ABAD'

The first pair shows an differential error result, which can be split into four matrices. When every column preceding MixColumn9 is injected with a fault, the matrices show as follows:

$$\begin{pmatrix} \text{'09'} & \text{'4C'} & \text{'60'} & \text{'B8'} \\ \text{'3B'} & \text{'B1'} & \text{'A5'} & \text{'1F'} \\ \text{'31'} & \text{'E6'} & \text{'36'} & \text{'A1'} \\ \text{'F1'} & \text{'8C'} & \text{'B5'} & \text{'50'} \end{pmatrix} = \begin{pmatrix} \text{'09'} & \text{'00'} & \text{'00'} & \text{'00'} \\ \text{'00'} & \text{'00'} & \text{'00'} & \text{'1F'} \\ \text{'00'} & \text{'00'} & \text{'36'} & \text{'00'} \\ \text{'00'} & \text{'8C'} & \text{'00'} & \text{'00'} \end{pmatrix} \oplus \begin{pmatrix} \text{'00'} & \text{'4C'} & \text{'00'} & \text{'00'} \\ \text{'3B'} & \text{'00'} & \text{'00'} & \text{'00'} \\ \text{'00'} & \text{'00'} & \text{'00'} & \text{'A1'} \\ \text{'00'} & \text{'00'} & \text{'B5'} & \text{'00'} \end{pmatrix}$$

$$\oplus \begin{pmatrix} \text{'00'} & \text{'00'} & \text{'60'} & \text{'00'} \\ \text{'00'} & \text{'B1'} & \text{'00'} & \text{'00'} \\ \text{'31'} & \text{'00'} & \text{'00'} & \text{'00'} \\ \text{'00'} & \text{'00'} & \text{'00'} & \text{'50'} \end{pmatrix} \oplus \begin{pmatrix} \text{'00'} & \text{'00'} & \text{'00'} & \text{'B8'} \\ \text{'00'} & \text{'00'} & \text{'A5'} & \text{'00'} \\ \text{'00'} & \text{'E6'} & \text{'00'} & \text{'00'} \\ \text{'F1'} & \text{'00'} & \text{'00'} & \text{'00'} \end{pmatrix}$$

By considering the first matrice, the possible values of the key can be reduced (for the first matrice, the error belongs to the first column). If the error lies in the first line: we have to compute the following

$$L_\varepsilon(1) = S_{2,\text{'09'}} \bigcap S_{1,\text{'1F'}} \bigcap S_{1,\text{'36'}} \bigcap S_{3,\text{'8C'}}$$

We have Card $L_\varepsilon(1) = 15$, hence, using $CipherText[0]$ ='46' and Proposition 6, we get the first set $PK_1[0]$ of values possibly taken on by $K_{10}[0]$.

If the error lies in the second line: we have to compute the following

$$L_\varepsilon(2) = S_{3,\text{'09'}} \bigcap S_{2,\text{'1F'}} \bigcap S_{1,\text{'36'}} \bigcap S_{1,\text{'8C'}}$$

We have Card $L_\varepsilon(2) = 15$, hence, using $CipherText[0]$ ='46' and Proposition 6, we get the second set $PK_2[0]$ of values possibly taken on by $K_{10}[0]$.

We go over the same steps in the cases where the error lies in the third or in the fourth line. We refer to the resulting sets by $PK_3[0]$ and $PK_4[0]$.

Finally, $K_{10}[0]$ lies in

$$PK_1[0] \cup PK_2[0] \cup PK_3[0] \cup PK_4[0]$$

which have 96 elements altogether. So we reduce the 256 values of $K_{10}[0]$ to only 96 possibilities.

We go over the same steps applied to the second, third and fourth matrices (which impact on the other bytes of K_{10}) and find

$$K_{10} = \text{'D014F9A8C9EE2589E13F0CC8B6630CA6'}$$

with ten faulty ciphertexts.

References

1. FIPS PUB 197 : *Avanced Encryption Standard*, `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`
2. Joan Daemen and Vincent Rijmen, *The Design of Rijndael, AES – The Advanced Encryption Standard*, Springer-Verlag 2002, (238 pp.).
3. Ross J. Anderson, Markus G. Kuhn: *Tamper Resistance – a Cautionary Note*, The Second USENIX Workshop on Electronic Commerce Proceedings, Oakland, California, November 18–21, 1996, pp 1–11, ISBN 1-880446-83-9.
4. E. Biham and A.Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, CS 0910, Proceedings of Crypto'97.
5. Boneh, DeMillo, and Lipton, *On the Importance of Checking Cryptographic Protocols for Faults*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'97, pp. 37–51, 1997.
6. Joan Daemen, *Annex to AES Proposal Rijndael*, `http://www.esat.kuleuven.ac.be/~rijmen/rijndael/PropCorr.PDF`,1998.
7. K. Nyberg, *Differentially uniform mappings for cryptography*, Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 55–64.
8. G. Letourneux, *Rapport de stage EDSI : Etude et implémentation de l'AES, Attaques DPA et DFA*, August 30, 2002.

Side-Channel Attack on Substitution Blocks

Roman Novak

Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia,
Roman.Novak@ijs.si

Abstract. We describe a side-channel attack on a substitution block, which is usually implemented as a table lookup operation. In particular, we have investigated smartcard implementations. The attack is based on the identifying equal intermediate results from power measurements while the actual values of these intermediates remain unknown. A powerful attack on substitution blocks can be mounted if the same table is used in multiple iterations and if cross-iteration comparisons are possible. Adversaries can use the method as a part of reverse engineering tools on secret algorithms. In addition to the described method, other methods have to be employed to completely restore the algorithm and its accompanying secret key. We have successfully used the method in a demonstration attack on a secret authentication and session-key generation algorithm implemented on SIM cards in GSM networks. The findings provide guidance for designing smartcard solutions that are secure against this kind of attack.

1 Introduction

Modelling cryptographic algorithms as mathematical objects cannot address weaknesses in the implementation of these algorithms in real-world cryptographic devices. Any real cryptographic device provides more information to a determined adversary than just the input plaintext and output ciphertext. This side-channel information is available as the timing of operations [1], power consumption of the devices [2], electromagnetic emanations [3], etc. Very little side-channel information is required to break many common ciphers. Non-invasive attacks and accompanying countermeasures have been studied extensively over the past few years.

Systems that rely on smartcards are of particular concern. Examples of such systems are secure Internet banking, remote access to corporate networks worldwide, existing GSM phone networks, pay-TV systems, electronic wallets etc. The embedded microcontroller accompanied with cryptoprocessor and memory capabilities promises numerous security benefits. However, as security processor technology advances, new techniques are developed that compromise the benefits of its use. Research on new attack techniques contributes to the improvement of future products.

Many supposedly secure implementations remain vulnerable. In this paper, a side-channel attack on substitution blocks is presented. Power measurements

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 307–318, 2003.

were used as the source of side-channel information. The substitution block is one of the fundamental primitives in cryptography for introducing non-linearity. In software, a substitution block is usually implemented as a table lookup operation. Obtaining resistance to side-channel attack appears to be a difficult task. The technique to be described can be used to attack implementations with weak side-channel countermeasures, i.e., inadequate implementation of masking techniques [4,5], or the implementation of secret algorithms. However, other methods should be employed to completely restore the secret algorithm and its accompanying key.

The side-channel attack on a substitution block is based on identifying equal intermediate results while the actual values of these intermediates remain unknown. An adversary can partially or fully restore the content of the lookup table and thus break an unknown substitution block. A demonstration attack was performed on the secret authentication and session-key generation algorithm implemented on SIM cards that are used in GSM networks. Specification of the secret algorithm was not available. All that was known was that the algorithm was not COMP128-1. Correctness of the restored algorithm was checked by means of plaintext/ciphertext pairs. A simplified example is given in order to keep the algorithm secret and be clear enough.

The purpose of this paper is to draw designers' attention to weak protective measures and to show that secret algorithms offer very little protection. The rest of the paper is structured as follows. Section 2 gives a short introduction to power analysis techniques. In Sect. 3 the problem of protecting substitution blocks is detailed. Some common errors in implementing the cardinal principle are given that can lead to the success of the proposed method. The environment in which the side-channel attack has been validated is described. The attack is presented in Sect. 4. The techniques that make search for lookup table feasible are given in Sect. 5 with examples. The actual attack would require the identification of relevant measurements. There are no general rules for identifying them reliably. Section 6 gives some guidelines that can be followed. Countermeasures against the proposed side-channel attack are discussed in Sec. 7. We conclude by summarising our findings in Sect. 8.

2 Power Analysis

Smart cards consist of logic gates, which are basically interconnected transistors. During operation, charges are applied to or removed from transistor gates. The sum of all charges can be measured through power consumption, on which power analysis techniques are based. A similar approach may be used on electromagnetic radiation traces.

Several variations of power analysis have been developed [2,6]. The power consumption measurements of smart card operations are interpreted directly in Simple Power Analysis (SPA). SPA can reveal hidden data in algorithms in which the execution path depends on the data being processed. More advanced techniques, like Differential Power Analysis (DPA) and Inferential Power Anal-

ysis (IPA), allow observation of the effects correlated to the data values being manipulated.

Power analysis attacks have been known for a while and effective countermeasures exist that pose difficulties, even to a well-funded and knowledgeable adversary [7]. On the other hand, it is difficult to address all the weaknesses in implementing a cryptographic algorithm. Frequently, the developers decide not to implement appropriate countermeasures if they believe that a particular power characteristic could not threaten the overall security scheme. This is not a practice to be followed.

Some of the countermeasures, like table mask operation [4,5], deal directly with the protection of substitution blocks; others protect the algorithm as a whole, i.e., timing randomness at the clock-cycle level [7]. Many countermeasures may be bypassed or compensated for [8]; the implementation of many others has security vulnerabilities. The side-channel attack on a substitution block may effectively remove such incomplete countermeasures. Furthermore, it can be used to reverse-engineer an unknown substitution block. As such, it is another argument against establishing secrecy by keeping cryptographic algorithms undisclosed.

3 Problem Description

Substitution has been known from traditional cryptography. For instance, one of the oldest known substitution ciphers is the Caesar cipher [9]. In this type of cipher each letter or group of letters is replaced by another letter or group of letters in order to disguise it. Modern cryptography still uses substitution as a building block in more complex compositions. By combining simple transformations it is possible to obtain strong ciphers. Substitution blocks are considered to provide a high security level because they contribute effectively to data diffusion.

Substitution block is a fundamental primitive used by many cryptographic algorithms such as DES, AES and COMP128. It is often implemented as table lookup. However, any implementation that directly looks up a table is vulnerable to differential side-channel attacks since the side-channel signals at the time of table lookup will correlate with each bit of the index accessed and with each bit of the value retrieved. To remove this correlation, protective measures have been proposed that are more or less close to the following cardinal principle [10]:

Definition 1 (cardinal principle). *Relevant bits of all intermediate cycles and their values should be statistically independent of the inputs, outputs and sensitive information if differential attacks are to be completely eliminated.*

For example, the table mask operation is proposed in [5]. For each instance of a cryptographic operation requiring one or more lookups of a table, a fresh random looking masked table is computed. This is done by selecting two permutations *ip* and *op* uniformly at random. The table indexes are then permuted using the permutation *ip*. Likewise, the table elements are permuted using the permutation *op*. A simple example would be to choose two random values x and

y, and then compute $T'(i \oplus x) = T(i) \oplus y$ for all values of index i, where T is the original table and T' is the masked table.

Implementation of such a protective measure, especially on devices with resource and cost limitations, is a challenging task. Many implementations fail to perform securely. The main problem with the table mask operation as described above is that it requires the table T' to be in RAM and the size of T' is the same as that of T. Moreover, the same table is used more than once in cryptographic algorithms. The use of the same masked table throughout the algorithm poses additional security threats, as an adversary may be able to restore some masked values using a method similar to the one described here.

Time limits on cryptographic algorithms prevent computation of a unique table for each use. For example, our secret authentication and session-key generation algorithm in GSM phone networks requires a lookup of two tables. Throughout the algorithm each table is used 1280 times. Therefore, 2560 table mask operations should be performed. In that case, the table mask operation itself may be the subject of an attack.

Some implementations use fixed masked tables in permanent memory that are specific to the card vendor or cryptographic device. For example, the size of the memory required to store the S-boxes in a masked version of the DES algorithm in [4] is too big for smartcards. The solution proposed is based on a secret bijective function which may be implemented as a fixed secret table. Other implementations may recalculate masked tables only during card initialisation, in which case different inputs may be fed to the algorithm with the same tables. Those solutions give little or no protection at all.

A fundamental rule of cryptography is that one must assume that the cryptanalyst knows the general method of encryption used. Experts have learned over the years that the only way to assure security is to follow an open design process, encouraging public review to identify flaws while they can still be fixed. Secrecy comes from having a strong public algorithm and a long key. However, many cryptographic algorithms are still kept secret. For instance, the GSM network operators use an updated version of COMP128-1, designated as COMP128-2, but the algorithm remains unpublished. Some network operators even develop a proprietary algorithm in secrecy. In either case, the algorithm used has not been publicly reviewed. A side-channel attack on the substitution block may be used as part of the reverse engineering tools against such secret algorithms.

The side-channel attack on substitution blocks described below has been validated on the SIM (Subscriber Identity Module) card deployed on several international networks. The contents of the lookup tables of the implemented authentication and session-key generation algorithm (A3A8) were not given to us. On the other hand, we knew the algorithm architecture, computations involved and the secret key. This information was restored using other reverse engineering tools, which are not covered in this paper. Some of the methods involved are new and will be published elsewhere. Protective measures against power analysis attacks were detected and compensated for. The card uses fixed tables; therefore, multiple repetitions at the same input can be used to aver-

age out the signal noise. The experiment was part of a larger ongoing project in which the significance of the side-channel information is evaluated in various reverse engineering techniques.

As described, the method requires from the cryptanalyst the ability to encrypt pieces of plaintext of his own choosing; however, it can be adapted to the situation where plaintext is known but not the ability to choose it.

4 Breaking a Substitution Block

Let $f(p)$ be a function that incorporates a lookup table T and some further transformations of the value read from the table. The parameter p represents plaintext input and may be extended to a sequence of parameters without significant change of the method.

$$r = f(p) \tag{1}$$

The problem that has to be solved by an adversary is to find the content of the unknown or modified lookup table T just by observing the side-channel information that is present in power variations. The value of parameter p is known to the attacker, while the result r is unknown, since it is further modified during algorithm execution.

The basic idea behind the attack on a substitution block is based on the fact that the same value of the parameter p gives the same intermediate result r, while different values of parameter p do not necessarily give different intermediate results r as soon as f is an injective function. By identifying equal intermediate results one can partially or fully restore the content of the lookup table and thus break an unknown substitution block. The method differs from template attack, where profiling of the experimental device is required before attack [11].

First, identification of the relevant measurements is needed to enable comparison of the results within algorithm execution by side-channel information alone. We define an equality function that is based on the relevant measurements.

Definition 2 (equality function). *Let N be the number of repetitions of the algorithm for a given value of parameter p that can be achieved within real-time constraints. Suppose $\overline{m}_{p,t}$ is the average of measurements of supply current $m_{p,t}$ at time t for a given value p,*

$$\overline{m}_{p,t} = \frac{1}{N} \sum_{j=1}^{N} m_{p,t} \ . \tag{2}$$

The equality function $ef(p_1, p_2, r)$ is defined such that $ef(p_1, p_2, r)$ is true if and only if $f(p_1)$ is equal to $f(p_2)$, where r are time indices $t_1, t_2, ..., t_n$ of the relevant measurements.

We use the following equality function in our experiments

$$ef(p_1, p_2, r) = \sum_{i=1}^{n} |\overline{m}_{p_1,t_i} - \overline{m}_{p_2,t_i}| < T \ , \tag{3}$$

where T is a threshold value. The function (3) does not necessarily exist for given smart card implementation. One can define different equality function for the same purpose of distinguishing values $f(p_1)$ and $f(p_2)$.

In the case of (3), the main problem for an adversary is to identify relevant measurements and to select suitable threshold value. Different techniques may be used for that purpose; several of them are mentioned in the following subsection. At this point we suppose that the equality function is known and can be evaluated effectively using a side-channel information leakage.

An equation can be written for each pair of parameters p_1 and p_2 for which the equality function holds:

$$f(p_1) = f(p_2), \ p_1 \neq p_2 \ . \tag{4}$$

It is not necessary to find all similarities and, hence, all equations. The set of equations is then solved for the table values with some degree-of-freedom, DF. The number of tables that solve the set of equations grows fast with DF. When the table contains a permutation, the number of solutions is bounded by the number of variations of 2^n elements choose DF, where each table entry is composed of n-bits. In case of a compression table, the upper bound on the number of solutions is 2^{nDF}. Nevertheless, only one solution to the set of equations is actually present in the implementation. An adversary can find the correct table by doing an exhaustive search. Therefore, it is important for DF to be very small, in practice not larger than 2. A large DF makes exhaustive search of the solution space infeasible. The right table can be identified using DPA on the computation results after the lookup operation.

5 Making the Search for Lookup Table Feasible

The requirement for small DF may prevent breaking a substitution block when only a single lookup operation is observed. However, the same table is frequently used several times within an algorithm because the same substitution block is used several times to form the product cipher. Several lookups at different positions can be observed and the resulting sets of equations combined together. Furthermore, when the same code is used for all these lookups, cross-comparisons of the results are possible. The set of equations may be extended further.

In Fig. 1 a computation is shown that includes four lookup operations. The computation can be found in the GSM authentication algorithm that has been deployed by different service providers in several types of SIM cards. Only a small fraction of the sixteen similar iterations is shown.

The algorithm is a keyed hash function. It takes a 16-byte key (128 bits) and 16-byte of data (128 bits) to output a 12-byte (96 bits) hash. The key k_0-k_{15}, as

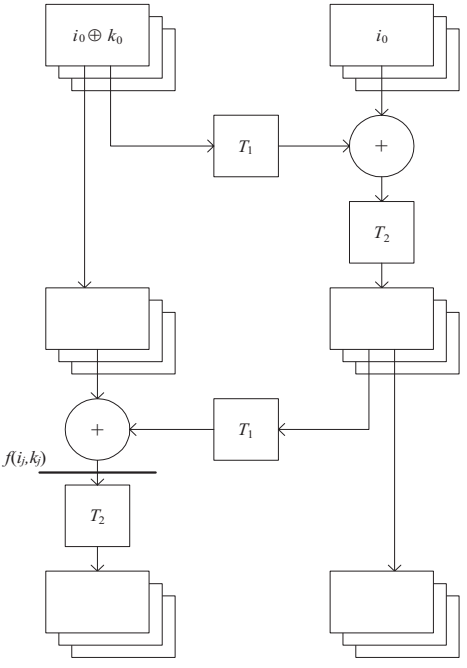

Fig. 1. Computation in the GSM authentication algorithm

used in the GSM protocol, is unique to each subscriber and is stored in the SIM card. The input data i_0-i_{15} is a random challenge supplied by the base station. The first 32 bits of the hash are used as a response to the challenge and sent back to the base station. The remaining 64 bits are used as a session key for voice encryption using the A5 algorithm.

In the example, the content of table T_1 is known while that of table T_2 is unknown. The relevant measurements have been identified such that (3) distinguishes different values $f(i_j, k_j)$, where $f(i_j, k_j)$ is defined as

$$f(i_j, k_j) = T_1(T_2(T_1(i_j \oplus k_j) \oplus i_j)) \oplus i_j \oplus k_j,\ j = 0...15\ . \tag{5}$$

The input value i_j may be chosen freely while the value k_j is part of the key, constant during investigation, and known to us. j is the iteration index. In the case of a single iteration, the resulting set of equations is highly undetermined, i.e. DF equals 157. Clearly, the restoration of table T_2 is an impossible task. However, the set of equations can be extended with equations from other iterations. Furthermore, cross-iteration comparisons are possible that reduce the number of iterations needed for a small DF. In order to demonstrate these two techniques, the 3-bit to 3-bit tables are used (Tab. 1), i.e., the tables consist of 2^3 elements of size 3-bits each.

The equality function can detect the following relations in the first iteration, where $k_0 = 5$:

Table 1. Lookup tables T_1 and T_2

index	0	1	2	3	4	5	6	7
T_1	2	5	3	6	1	7	4	0
T_2	7	1	3	0	6	2	5	4

$$\begin{gathered} f(0,5) = f(1,5) = f(4,5) = 4 \\ f(2,5) = f(5,5) = 1 \ . \end{gathered} \tag{6}$$

The actual function values are shown just to make the example more illustrative. They are unknown to an adversary during a real attack. The function values for inputs 3, 6 and 7 differ from each other and are not shown.

The next step is to write a set of equations based on the similarities. In this case, the degree-of-freedom equals 5, since there are 8 variables $T_2(x)$, and 4 of them are related to each other as follows:

$$\begin{gathered} T_1(T_2(7)) \oplus 5 = T_1(T_2(0)) \oplus 4 = T_1(T_2(1)) \oplus 1 \\ T_1(T_2(2)) \oplus 7 = T_1(T_2(7)) \ . \end{gathered} \tag{7}$$

Analysis of the second iteration with $k_1 = 7$ gives the relations

$$\begin{gathered} f(6,7) = f(7,7) = 3 \\ f(1,7) = f(4,7) = f(5,7) = 5 \end{gathered} \tag{8}$$

and a set of equations

$$\begin{gathered} T_1(T_2(3)) \oplus 1 = T_1(T_2(5)) \\ T_1(T_2(5)) \oplus 6 = T_1(T_2(2)) \oplus 3 = T_1(T_2(6)) \oplus 2 \ . \end{gathered} \tag{9}$$

Again, DF equals 5; however, when (7) and (9) are merged, the resulting DF equals 2. The process can be continued with other iterations until a DF of 1 is achieved, which is the lower bound on DF in this example. We applied this technique on our A3A8 algorithm that uses an 8-bit to 8-bit table T_2. The equations from 8 iterations out of 16 had to be merged in order to reach DF of 1. The actual number of iterations may vary with key values.

When the same code performs computations within iterations with minor differences in power consumption patterns, the possibility of finding relevant measurements exists such that cross-iteration comparisons can be carried out. In that case, the definition of the equality function has to be changed slightly in order to compare measurements with different time indices. In our simplified example, cross-iteration comparisons would produce the following relations, in addition to the relations stated so far:

$$\begin{gathered} f(3,5) = f(0,7) = 7 \\ f(6,5) = f(6,7) = f(7,7) = 3 \\ f(7,5) = f(2,7) = 6 \ . \end{gathered} \tag{10}$$

The final set of equations would contain (7), (9), and the following equations:

$$\begin{gathered} T_1(T_2(7)) \oplus 6 = T_1(T_2(0)) \oplus 7 \\ T_1(T_2(0)) \oplus 3 = T_1(T_2(3)) \oplus 1 = T_1(T_2(5)) \\ T_1(T_2(4)) \oplus 2 = T_1(T_2(5)) \oplus 5 \ . \end{gathered} \tag{11}$$

It can be shown that this set of equations has DF of 1. One must just pick a value for arbitrary table location and compute the rest of the table. 2^3 different values may be selected; the right one should be confirmed by an alternative method, for instance, a DPA on intermediate results after the lookup operation. We managed to identify relevant measurements in power traces for our SIM card that allowed cross-iteration comparisons. A power trace refers to a set of power consumption measurements taken across a card operation. Only 4 iterations were needed to achieve DF of 1 in the real world example.

6 Identification of Relevant Measurements

Clearly, the actual attack would be highly dependent on the algorithm being implemented and the architecture being used, and would require some guesswork on the part of the attacker as to the relevant measurements and the types of software countermeasures being used.

Identification of relevant measurements is a prerequisite for the success of the method. Relevant measurements are strongly correlated to the value $f(p)$. Such measurements can be taken during the processing of the value $f(p)$ and its transforms. For instance, the value $f(p)$ may be used as an index into the next lookup table T. As long as the table implements permutation, it preserves the equality relation, $T(f(p_1)) = T(f(p_2)) \Leftrightarrow f(p_1) = f(p_2)$. The measurements that are taken while $T(f(p))$ is read from or written to the memory may contribute to the existence of the equality function, and can, as such, be considered as relevant measurements.

Many measurements may contain effects correlated to the $f(p)$; not all of them are relevant measurements. For instance, Hamming weight information, which can be leaked during a bus transfer of parameter p [12], is usually weakly correlated to the $f(p)$. However, the averages $\overline{m}_{p_1,t_i}$ and $\overline{m}_{p_2,t_i}$, where $f(p_1) = f(p_2)$, may differ too much to include time index t_i in (3).

Some features of a power trace can be helpful in identifying relevant measurements. Such a feature can be a power pattern during memory access. Table lookups are usually implemented as memory read operations. They can be identified easily unless proper countermeasures are present. In Fig. 2 a power trace is shown for the first two lookups on our test SIM card. Power consumption measurements were obtained by measuring voltage variations across a resistor (25 ohm) that was inserted in series with the card ground pin. The sampling speed was set at approximately 7.15 MSamples/s. 14-bit resolution was used.

Underlined patterns clearly differ from other controller activities. Firstly, supply current increases; we suppose that this is due to the memory addressing phase. After that, a delay is introduced with lower power consumption before actual data retrieval. The measurements in the second group are the relevant

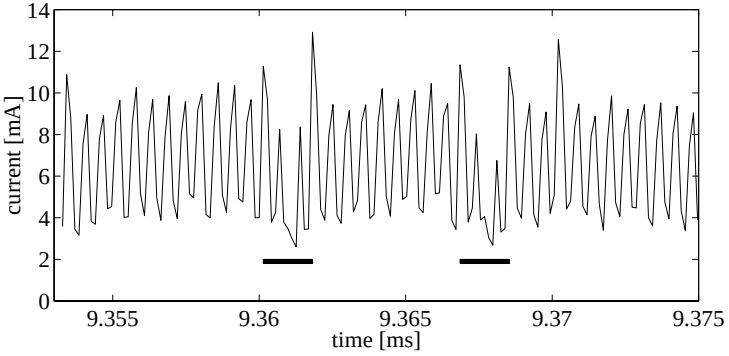

Fig. 2. Power trace of two lookup operations on a test SIM card

measurements that can be used for restoring the content of the first lookup table. The measurements at the beginning of the selected interval contain effects correlated to the value $f(p)$, while the measurements at the end correlate to the value $T_2(f(p))$.

Other methods may be used to identify relevant measurements when they are not so obvious. The methods are based mainly on various correlation techniques. However, there is no general rule for identifying them reliably. The only confirmation that a proper equality function has been selected is the success of the method. Although the feasibility of the attack on real cards seams low due to probability of errors in the equality test, the existence of multiple relevant measurements and the possibility to average out noise usually lead to the success of the method.

7 Countermeasures

Many authors provide guidance for designing smart card solutions against power analysis attacks. Techniques for preventing side-channel attack on substitution blocks, as described in this paper, fall roughly into two categories.

A first approach is to prevent information leakage, using the general techniques that protect the algorithm as a whole. Well-known techniques are signal size reduction, introducing noise into power consumption, randomised execution timing and order, balancing state transitions, key use counters, physically shielding the device, blinding, multithreaded applications, etc [13,2,7,10]. All these techniques make identification of relevant measurements difficult or even impossible.

On the other hand, many techniques can be used to bypass or compensate for these countermeasures. We suggest the use of as many redundant countermeasures as the available resources permit, because many of the countermeasures can be compensated for if they are implemented alone [8]. In our experience, a subset of countermeasures offers a higher level of protection. For instance, attacks on substitution blocks, and many other power analysis techniques, require

the attacker to predict the time at which a certain instruction is executed. As a protection, the designers insert random-time delays between the observable reactions that may be the subject of the attack. This countermeasure can be overcome rather easily using correlation techniques. The presence of other countermeasures, like feeding registers and busses with random values, can significantly strengthen the protection. The next example of synergic countermeasures is noise introduction in combination with key use counters. Noise can usually be averaged out, but in that case, a counter can prevent the attacker from gathering large number of samples.

A second category of countermeasures against side-channel attack on substitution blocks involves measures that are related to the content of the lookup table. The best way to eliminate the attack is adherence to the cardinal principle [10]. The table mask operation [5] is such a solution when different tables are used for each access. Great care must be taken when masking is performed in order not to create new security vulnerabilities.

8 Conclusion

The level of tamper resistance offered by any particular product can be measured by the time and cost penalty that the protective mechanisms impose on the attacker. The realities of a physical implementation can be extremely difficult to control. When the method is known, a side-channel attack on moderately protected smartcard typically requires a few minutes to several hours to complete, while the cost of the sampling equipment falls in the range from several hundred to several thousand dollars.

We have introduced a side-channel attack on substitution blocks, a fundamental primitive used by many cryptographic algorithms. We have shown how side-channel information could be used effectively on implementations that have been equipped with ad hoc and inadequate countermeasures. Although those countermeasures may resist some side-channel attacks, they fail to adhere to the cardinal principle. The method can be used as a part of reverse-engineering tools in the attacks on unknown algorithms.

The unknown intermediate results within the execution of the algorithm are compared. The comparisons form the basis for a set of equations, which is solved for the table values. An even more powerful attack on substitution blocks can be mounted if the table is used in multiple iterations and when cross-iteration comparisons are possible. The identification of relevant measurements is a prerequisite for the success of the method. We have shown how some features of a power trace help in identifying relevant measurements.

Proper countermeasures make identification of relevant measurements difficult or even impossible. The use of redundant countermeasures is suggested, since many protective measures can be bypassed or compensated for. The best way to eliminate attack is strict adherence to the cardinal principle. The designers must not rely on secrecy of the algorithm, as the algorithm may be reverse-engineered in the presence of side-channel information leakage.

References

1. Kocher, P.: Timing Attacks on Implementation of Diffie-Hellman, RSA, DSS and Other Systems. In: Koblitz, N. (ed.): Advances in Cryptology – Crypto'96. Lecture Notes in Computer Science, Vol. 1109. Springer-Verlag, Berlin Heidelberg New York (1996) 104–113
2. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.): Advances in Cryptology – Crypto'99. Lecture Notes in Computer Science, Vol. 1666. Springer-Verlag, Berlin Heidelberg New York (1999) 388–397
3. Agrawal D., Archambeault B., Rao J.R., Rohatgi P.: The EM Side-Channel(s): Attacks and Assessment Methodologies. In: Cryptographic Hardware and Embedded Systems – CHES'2002
4. Goubin L., Patarin J.: DES and Differential Power Analysis. In. Koc C.K., Paar C. (ed.): Cryptographic Hardware and Embedded Systems – CHES'1999. Lecture Notes in Computer Science, Vol. 1717. Springer-Verlag, Berlin Heidelberg New York (1999) 158–172
5. Rao J.R., Rohatgi P., Scherzer H., Tinguely S.: Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards. Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, California, May 12–15, IEEE Computer Society (2002) 31–44
6. Fahn, P.N., Pearson, P.K.: IPA: A New Class of Power Attacks. In: Koc, C.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES'99. Lecture Notes in Computer Science, Vol. 1717. Springer-Verlag, Berlin Heidelberg New York (1999) 173–186
7. Kömmerling, O., Kuhn, M.G.: Design Principles for Tamper-Resistant Smartcard Processors. Proceedings of the USENIX Workshop on Smartcard Technology – Smartcard'99, Chicago, Illinois, May 10–11, USENIX Association (1999) 9–20
8. Clavier C., Coron J.S., Dabbous N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In. Koc C.K., Paar C. (ed.): Cryptographic Hardware and Embedded Systems – CHES'2000. Lecture Notes in Computer Science, Vol. 1965. Springer-Verlag, Berlin Heidelberg New York (2000) 252–263
9. Tanenbaum A.S.: Computer Networks. Prentice Hall PTR (2002)
10. Chari S., Jutla C.S., Rao J.R., Rohatgi P.: Towards Sound Countermeasures to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.): Advances in Cryptology – Crypto'99. Lecture Notes in Computer Science, Vol. 1666. Springer-Verlag, Berlin Heidelberg New York (1999) 398–412
11. Chari S., Rao J.R., Rohatgi P.: Template Attacks. In: Cryptographic Hardware and Embedded Systems – CHES'2002
12. Messerges T.S., Dabbish E.A., Sloan R.H.: Examining Smart-Card Security under the Threat of Power Analysis Attacks. IEEE Transactions on Computers, **51(5)** (2002) 541–552
13. Anderson, R., Kuhn, M.: Low Cost Attacks on Tamper Resistant Devices. In: Lomas, M. et al. (ed.): Security Protocols. Lecture Notes in Computer Science, Vol. 1361. Springer-Verlag, Berlin Heidelberg New York (1997) 125–136

Timing Attack against Implementation of a Parallel Algorithm for Modular Exponentiation

Yasuyuki Sakai[1] and Kouichi Sakurai[2]

[1] Mitsubishi Electric Corporation,
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan
ysakai@iss.isl.melco.co.jp
[2] Kyushu University,
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. We describe a parallel algorithm for modular exponentiation $y \equiv x^k \bmod n$. Then we discuss timing attacks against an implementation of the proposed parallel algorithm for modular exponentiation. When we have two processors, which perform modular exponentiation, an exponent k is scattered into two partial exponents $k^{(0)}$ and $k^{(1)}$, where $k^{(0)}$ and $k^{(1)}$ are derived by bitwise AND operation from k such that $k^{(0)} = k \wedge (0101 \cdots 01)_2$ and $k^{(1)} = k \wedge (1010 \cdots 10)_2$. Two partial modular exponentiations $y_0 \equiv x^{k^{(0)}} \bmod n$ and $y_1 \equiv x^{k^{(1)}} \bmod n$ are performed in parallel using two processors. Then we can obtain y by computing $y \equiv y_0 y_1 \bmod n$. In general, the hamming weight of $k^{(0)}$ and $k^{(1)}$ are smaller than that of k. Thus fast computation of modular exponentiation $y \equiv x^k \bmod n$ can be achieved. Moreover we show a timing attack against an implementation of this algorithm. We perform a software simulation of the attack and analyze security of the parallel implementation.

Keywords: Parallel modular exponentiation, Montgomery multiplication, Side channel attack, Timing attack, RSA cryptosystems

1 Introduction

1.1 Timing Attack

Smart cards are one of the major application fields of cryptographic algorithms, and may contain sensitive data, such as RSA private key. Some implementations of cryptographic algorithms often leak "*side channel information*." Side channel information includes power consumption, electromagnetic fields and timing to process. Side channel attacks, which use side channel information leaked from real implementation of cryptographic algorithms, were first introduced by Kocher [Ko96,KJJ99]. Side channel attacks can be often much more powerful than mathematical cryptanalysis. Thus, many literatures on side channel cryptanalysis have been published [IT02,IYTT02,OS00,Sc00,Wa99,WT01].

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 319–330, 2003.

In this paper, we focus on a timing attack against an implementation of a parallel algorithm for modular exponentiation which will be described in later section. The running time of a cryptographic device can constitute an information channel, providing the attacker with valuable information on the secret parameters involved. The timing attack is the attack to determine a secret parameter from differences between running times needed for various input values. The timing attack was first introduced by Kocher at Crypto 96 [Ko96]. He showed that a careful statistical analysis could lead to the total recovery of secret parameters.

In [DKLMQW98] a successful timing attack against a modular exponentiation without Chinese Remainder Theorem (CRT) was implemented. In [SQK01] further improved timing attacks were proposed. These attacks assume that implementations do not use CRT for modular exponentiation. In [Sc00] Schindler presented a powerful timing attack against an implementation using CRT. The exponent of modular exponentiation can be a secret parameter in RSA-based cryptosystems. Thus we must implement modular exponentiation very carefully.

In RSA-based cryptosystems, modular exponentiation is the most expensive part in implementation. Therefore, it is very attractive to provide algorithms that allow efficient implementation of modular exponentiation. Montgomery's method [Mo85] perform modular multiplication without a division instruction which is an expensive operation in almost processors. Thus this method can achieve computational speed-up and is often used in implementations of modular exponentiation. Timing attacks described in the above literature were against modular exponentiation with Montgomery's method. The running time of Montgomery's method for modular multiplication depends on the input value. If the intermediary result of the multiplication is greater than the modulus, an "*extra subtraction*" has to be performed. The extra subtraction lead differences between running times needed for various input values. Previous works on timing attacks against the modular exponentiation [DKLMQW98,SQK01,Sc00] made good use of the timing differences. Dhem et. al. presented a practical attack such that a secret exponent of 512 bits can be totally recovered with 300,000 time measurements [DKLMQW98]. In their attacks, an attacker has to know information on the implementation such that modular exponentiation is implemented with the binary method and Montgomery's multiplication. Walter et. al. gave a way to recover a secret exponent without knowledge of the modulus or input values [WT01].

1.2 Parallel Algorithm

Recently Garcia et. al. gave parallel algorithms for elliptic scalar multiplication [GG02]. When we have plural devises, which can perform elliptic scalar multiplications, we scatter a given scalar into several sub-scalars. Partial scalar multiplication are carried out with sub-scalar in parallel on plural devises. Then scalar multiplication can be computed with the outputs from the devises. Nonzero bits in scalar representation can be distributed into plural devises. Thus this

parallel algorithm provides an efficient implementation. This algorithm for elliptic scalar multiplication can be easily extended to modular exponentiation. In this paper we show a parallel algorithm for modular exponentiation. Our parallel algorithm can achieve speed-up of computation. However, as we noted before, computational advantage can not be enough for implementations, especially on smart cards. That is, side channel attacks have to be cared when we implement our parallel algorithm.

1.3 Our Contribution

In this paper we first describe a parallel algorithm for modular exponentiation. Then the computational efficiency will be discussed. Our parallel algorithm can gain speed in modular exponentiation if we have plural devises for modular exponentiation. When we have b processors, the computational complexity can be reduced to $(t-1)\mathcal{S} + (\lceil H(k)/b \rceil + b - 1)\mathcal{M}$ on the best case, where t and $H(k)$ denote the bitlength of k and hamming weight of k respectively, $\mathcal{S}$ and $\mathcal{M}$ denote modular squaring and multiplication respectively.

Moreover we analyze vulnerability of implementations of our parallel algorithm against the timing attack. We will state that there is difficulty in the attack against our parallel implementation. We adopt Dhem's strategy [DKLMQW98] based timing attack. We make an experiment on the timing attack with software simulation. Our results show that Dhem's method based attack can still work against the parallel implementation.

2 Parallel Algorithm for Modular Exponentiation

In this section we describe a parallel algorithm for modular exponentiation $y \equiv x^k \bmod n$. Garcia et. al. [GG02] gave a parallel algorithm for elliptic scalar multiplication, which is the basis of our algorithm.

2.1 The Algorithm

Assume that a t-bit non negative integer k has a binary representation $k = (k_{t-1} \cdots k_1 k_0), k_i \in \{0,1\}$. We divide the binary representation of k in $\lceil t/b \rceil$ blocks of b bits each one and then we scatter k into $k^{(0)}, k^{(1)}, \cdots, k^{(b-1)}$. The binary representation of the block $k^{(i)}$ is formed by $\lceil t/b \rceil$ blocks of b bits set to 0, except for the i-th bit, which has the same value that the i-th bit of the corresponding block of the binary representation of k. Thus we can define $k^{(i)}$, for $i = 0, \cdots, b-1$, as follows.

$$
\begin{aligned}
k^{(0)} &= k_0 + k_b 2^b + k_{2b} 2^{2b} + \cdots + k_{(\lceil \frac{t}{b} \rceil - 1)b} 2^{(\lceil \frac{t}{b} \rceil - 1)b} \\
k^{(1)} &= k_1 2 + k_{b+1} 2^{b+1} + k_{2b+1} 2^{2b+1} + \cdots + k_{(\lceil \frac{t}{b} \rceil - 1)b+1} 2^{(\lceil \frac{t}{b} \rceil - 1)b+1} \\
&\vdots \\
k^{(b-1)} &= k_{b-1} 2^{b-1} + k_{2b-1} 2^{2b-1} + k_{3b-1} 2^{3b-1} + \cdots + k_{b\lceil \frac{t}{b} \rceil - 1} 2^{b\lceil \frac{t}{b} \rceil - 1}
\end{aligned}
$$

That is

$$k^{(i)} = \sum_{j=0}^{\lceil \frac{t}{b} \rceil - 1} k_{jb+i} 2^{jb+i}$$

for $i = 0, 1, \cdots, b-1$, where we consider some padding bits $b_l = 0$ for $t - 1 < l < b\lceil t/b \rceil - 1$.

Clearly we have

$$k = k^{(0)} + k^{(1)} +, \cdots, +k^{(b-1)}.$$

Thus we can compute the modular exponentiation $y \equiv x^k \bmod n$ by the following equation.

$$x^k \bmod n = \left(\left(x^{k^{(0)}} \bmod n \right) \left(x^{k^{(1)}} \bmod n \right) \cdots \left(x^{k^{(b-1)}} \bmod n \right) \right) \bmod n$$

The exponent set $\{k^{(0)}, k^{(1)}, \cdots, k^{(b-1)}\}$ can be easily obtained from k by bitwise AND operation with the appropriate mask.

We show the algorithm for parallel modular exponentiation in Algorithm 1.

Algorithm 1 Parallel modular exponentiation

`Input` x, n, k
`Output` $y \equiv x^k \bmod n$

1. `Bits scattering:`
   Compute the set $\left\{ k^{(0)}, k^{(1)}, \cdots, k^{(b-1)} \right\}$
2. `Parallel computation:`
   Using b processors, compute in parallel the exponentiations

$$y \equiv \left(\left(x^{k^{(0)}} \bmod n \right) \left(x^{k^{(1)}} \bmod n \right) \cdots \left(x^{k^{(b-1)}} \bmod n \right) \right) \bmod n$$

3. `Return` y

Notice that in the case that block-length b equals to one, Algorithm 1 is equivalent to the simple modular exponentiation.

2.2 Computational Complexity

Let we discuss the computational complexity of the parallel modular exponentiation Algorithm 1. Assume k has the bitlength t and has the hamming weight $H(k)$. Then we have the following theorem [GG02].

Theorem 1. *Assume each individual exponentiation is performed by the binary method (Algorithm 2). The parallel exponentiation Algorithm 1 has the computational complexity, on the best case,*

$$(t-1)\,\mathcal{S} + (\lceil H(k)/b \rceil + b - 1)\,\mathcal{M}$$

and, on the worst case,

$$(t-1)\,\mathcal{S} + (H(k)-1)\,\mathcal{M}$$

where $\mathcal{M}$ and $\mathcal{S}$ denote modular multiplication and modular squaring, respectively.

The computational complexity of the parallel exponentiation can be evaluated by the individual exponentiation which has the most expensive complexity. Since the most significant bit k_{t-1} is 1 by assumption, one of the individual exponentiation has to perform $t-1$ modular squarings.

On the best case, all the bits set to one in the binary representation of k are equally scattered among the exponent set $\{k^{(i)}\}$, so the computational cost is perfectly balanced on all the individual exponentiation.

The worst case implies that there exists some index i, for $0 \leq i \leq b-1$, such that all of bit "1" are mapped to $k^{(i)}$ and then $x^{k^{(i)}} \bmod n = x^k \bmod n$. In such the case, the computational cost is the same as the traditional binary exponentiation.

The computational cost for computing the set $\{k^{(0)}, k^{(1)}, \cdots, k^{(b-1)}\}$ from k can be ignored, because we can obtain the set by simple bitwise AND operations.

3 Timing Attack against Modular Exponentiation

In this section we explain a framework of the timing attack against implementation of the basic modular exponentiation without CRT.

3.1 Modular Exponentiation

In this paper we consider the binary representation for a given exponent and the left-to-right binary (square-and-multiply) method for modular exponentiation. The left-to-right binary method for modular exponentiation $y \equiv x^k \bmod n$ can be described as Algorithm 2.

Algorithm 2 Left-to-right binary method of modular exponentiation

```
Input  x, n, k, where k = (k_{t-1} k_{t-2} ... k_0), k_i ∈ {0,1} for 0 ≤ i ≤ t-2 and
       k_{t-1} = 1
Output y ≡ x^k mod n
1.  y ← x
2.  for i from t-2 downto 0
        y ← y^2 mod n
        if k_i = 1 then
            y ← y · x mod n
    endfor
3.  return y
```

The timing attack will be demonstrated on an implementation with the following algorithms.

- The modular exponentiation is performed by Algorithm 2.
- The modular exponentiation is performed without CRT.
- Modular multiplication and modular squaring are performed with Montgomery's method.

In the Montgomery's method, when the intermediate value during the computation becomes larger than the modulus n, we have to perform "*extra subtraction*."

The goal for an attacker is to recover the exponent k, which can be a secret parameter of the decryption and the signature generation in RSA-based cryptosystems. The attacker determine a secret parameter k from differences between running times needed for various input values x.

3.2 Model of the Attacker

We assume that the attacker can be modeled as the following.

- The attacker has access to a device which perform modular exponentiation with a secret exponent.
- The attacker can measure the running time of the total (not partial) modular exponentiation with various input x.
- The attacker knows the modulus n.
- The attacker has the knowledge of the implementation. In this paper, we assume that the modular exponentiation is performed by the method described in the previous section.

The attack algorithm will be developed with the assumption that the running time of modular exponentiation only depend on the base x but not on other influences.

3.3 Dhem's Method of Timing Attack

Dhem et. al. proposed a practical timing attack against an implementation of modular exponentiation without CRT [DKLMQW98]. In this subsection we briefly explain their strategy. See [DKLMQW98] for details. Our timing attack, which will be described in the next section, will be developed based on their attack.

Assume again that a given exponent k has binary representation $k = (k_{t-1} \cdots k_1 k_0)$, where $k \in \{0, 1\}, k_{t-1} = 1$. Their attack recovers the exponent bit by bit from k_{t-2} to the least significant bit k_0. Notice that the MSB k_{t-1} is always 1. We start by attacking k_{t-2}. When $k_{t-2} = 1$, at `Step 2` of Algorithm 2, modular multiplication with Montgomery's method has to be performed. For some input x, intermediate value can be larger than the modulus n, and then the extra subtraction has to be performed. For the other input x, the extra subtraction

is not required. Let X be the set of inputs. We can define two subsets of inputs $X_1, X_2 \subset X$ as follows.

$$X_1 = \{x \in X | x \cdot x^2 \text{ has to be performed with extra subtraction}\}$$
$$X_2 = \{x \in X | x \cdot x^2 \text{ can be performed without extra subtraction}\}$$

If the value of k_{t-2} is 1, then we can expect that the running times for the inputs $x \in X_1$ to be slightly higher than the corresponding times for $x \in X_2$.

On the other hand, if the actual value of k_{t-2} is 0, then the modular multiplication in `Step 2` will not be performed. In this case, for any input x, there is no reason that the extra subtraction is induced. Therefore, the separation in two subsets should look random, and we should not observe any significant differences in the running time.

When the attacker wants to guess the bit k_{t-2}, he should take the strategy below.

Algorithm 3 Guessing k_{t-2}

---

1. `Generating two subsets` X_1, X_2`:`
   For various inputs x, the attacker does the following simulation with the knowledge of the implementation. At modular multiplication phase in `Step 2` of Algorithm 2 with $i = t-2$, if the extra subtraction has to be performed, x should be classified into X_1. Else if the extra subtraction is not required, x should be classified into X_2.
2. `Measuring the running times:`
   Using the device, on which modular exponentiation is implemented, the attacker measures the running time of modular exponentiation for $x \in X_1$ and $x \in X_2$.
3. `Guessing` k_{t-2}`:`
   The attacker does a statistical analysis on the difference of the running times between $x \in X_1$ and $x \in X_2$. Then he guesses $k_{t-2} = 0$ or 1.

Based on the time measurement, the attacker has to decide that the two subsets X_1 and X_2 are significantly different or not. Some statistical analysis can be of help. Possible use for statistics could be the mean value and χ^2 test.

The attacker first guesses the bit k_{t-2} based on Algorithm 3. The same strategy can be applied continuously bit by bit from MSB to LSB. The attacker may recover the total secret exponent k.

There is a more subtle way to take advantage of our knowledge of Montgomery's method: instead of the multiplication phase, we could turn ourselves to the square phase at `Step 2` of Algorithm 2 [DKLMQW98]. The same strategy as multiplication phase described before can be applicable.

4 A Timing Attack against the Parallel Modular Exponentiation

In this section we consider a timing attack against the parallel algorithm for modular exponentiation Algorithm 1. Two parallelized exponentiation will be discussed.

4.1 The Difficulty

The total running time of the parallel algorithm for modular exponentiation depends on the most low-speed partial exponentiation among $x^{k^{(0)}} \bmod n$, $x^{k^{(1)}} \bmod n, \cdots, x^{k^{(b-1)}} \bmod n$. This property causes difficulty such that the running time of a cryptographic device could not constitute an information channel on all bits of k.

Let we consider the case of two parallelism. In that case two partial exponents $k^{(0)}$ and $k^{(1)}$ should be derived from the given exponent k as below.

$$k^{(0)} = k \wedge (0101 \cdots 01)_2$$
$$k^{(1)} = k \wedge (1010 \cdots 10)_2$$

where $\wedge$ denotes bitwise AND operation.

The computational complexity of the partial modular exponentiations using left-to-right method of modular exponentiation Algorithm 2 can be evaluated as

$$(t_i - 1)\mathcal{S} + \left(H(k^{(i)}) - 1\right)\mathcal{M} \tag{1}$$

for $i = 0, 1$, where t_i denotes the bitlength of $k^{(i)}$. Note that $k^{(1)}$ always has bitlength t. The hamming weight $H(k^{(i)})$ of the partial exponents $k^{(i)}$ has significant effect on the running time of the total modular exponentiation.

In the next subsection we will discuss this effect.

4.2 The Case That Hamming Weight of $k^{(0)}$ and $k^{(1)}$ Are almost the Same

We can state the following theorem.

Theorem 2. *Assume that the running time of modular exponentiation can be evaluated by the number of modular multiplication and squaring required, that is, other influences can be ignored. Let $k^{(0)}$ and $k^{(1)}$ are derived from k by masking as before. If the following equation holds, the running time of the two partial modular equations $x^{k^{(0)}} \bmod n$ and $x^{k^{(1)}} \bmod n$ have almost the same running time.*

$$H(k^{(0)}) - \text{“the run-length of the leading bit 0 in } k^{(0)}\text{”} - 1 = H(k^{(1)}) - 1 \tag{2}$$

Proof. Notice that the MSB of $k^{(1)}$ is always 1. The evaluation (2) can be easily derived from (1). □

Following is a small example.

$$\begin{aligned} k &= 1\,1\,1\,1\cdots 1\,1\,0\,1 \\ k^{(0)} &= 0\,1\,0\,1\cdots 0\,1\,0\,1 \\ k^{(1)} &= 1\,0\,1\,0\cdots 1\,0\,0\,0 \end{aligned}$$

In this case, the running time of $x^{k^{(0)}} \bmod n$ and $x^{k^{(1)}} \bmod n$ can be almost the same.

For randomly chosen input x, The running time of the total exponentiation $x^k \bmod n$ could be an information on $k^{(0)}$ and $k^{(1)}$.

4.3 The Case That the Difference between Hamming Weight of $k^{(0)}$ and $k^{(1)}$ Is Large

When $H(k^{(0)})$ is significantly larger than $H(k^{(1)})$, the running time of the total modular exponentiation $x^k \bmod n$ can be regarded as that of $x^{k^{(0)}} \bmod n$. Therefore, in this case any information on $k^{(1)}$ can not be leaked in the running time of total modular exponentiation. Then the attacker has little chance to recover $k^{(0)}$ from the running time.

When the relation between $H(k^{(0)})$ and $H(k^{(1)})$ is far from the evaluation (2) in theorem 2, how better chance for successful attack the attacker can get may depend on the following.

- The ratio of the running time of the extra subtraction to the running time of $x^{k^{(0)}} \bmod n$.
- The influences on running time other than Montgomery's method of modular multiplication and squaring.

4.4 An Algorithm for Attack against the Parallel Implementation

In this subsection we state an algorithm for timing attack against our parallel algorithm of modular exponentiation Algorithm 1. We consider the two parallelized implementation.

Similar to the attack against traditional modular exponentiation, described in the previous section, we define the model of the attacker as follows.

- The attacker has access to a device which perform the modular exponentiation with a secret exponent.
- The attacker can measure the running time of the total (not partial) modular exponentiation with various input x.
- The attacker knows the modulus n.
- The attacker has the knowledge of the implementation such that:
  - The modular exponentiation is performed by the two parallelized algorithm (i.e. $b = 2$).

- The each partial modular exponentiation is performed by left-to-right binary method.
- The partial exponentiations are performed without CRT.
- Modular multiplication and modular squaring are performed with Montgomery's method.

We assume again that the MSB k_{t-1} of the secret exponent k is always 1. We also assume that the MSB of $k^{(1)}$ is always 1 by appropriate masking. The strategy for guessing k_i is quite similar to Algorithm 3. Algorithm 4 shows the strategy to guess the second significant bit k_{t-2} of the given exponent k.

Algorithm 4 Guessing k_{t-2} in the two parallelized implementation

1. `Generating two subsets` X_1, X_2`:`
   For various inputs x, the attacker does the following simulation with the knowledge of the implementation. At modular multiplication phase in Algorithm 1 with $i = t-2$, if the extra subtraction has to be performed, x should be classified into X_1. Else if the extra subtraction is not required, x should be classified into X_2.
2. `Measuring the running times:`
   Using the device, on which modular exponentiation is implemented, the attacker measures the running time of modular exponentiation for $x \in X_1$ and $x \in X_2$.
3. `Guessing` k_{t-2}`:`
   The attacker does a statistical analysis on the difference of the running times between $x \in X_1$ and $x \in X_2$. Then he guesses $k_{t-2} = 0$ or 1.

The attacker first guesses the bit k_{t-2} based on Algorithm 4. The same strategy can be applied continuously bit by bit from MSB to LSB. The attacker may recover the total secret exponent k.

5 An Experiment

In this section we show an experiment on the attack to recover the secret exponent k. As in the previous section, we will consider the two parallelized implementation of our Algorithm 1 (i.e. $b = 2$). We made a software simulation on Pentium IV-based PC, running at 1.8 GHz. Programs were written in C-language with VC++ 6.0 compiler.

In this environment, when the modulus n and the exponent k has the size of 128 bits, the mean value of clock cycles needed to perform extra subtraction was several hundreds. We used the mean value for statistical analysis as follows.

– For guessing k_i, if the difference of the mean value of the running time between input $x \in X_1$ and input $x \in X_2$ is larger than several hundreds clock cycles, then the attacker should guess $k_i = 1$.

– If the difference is smaller than several hundreds clock cycles, then the attacker should guess $k_i = 0$.

In the case that both n and k has 128 bits size, when we took 100,000 time measurements, the following results were obtained by our experiment.

– When the relation between two partial exponents $k^{(0)}$ and $k^{(1)}$ almost holds equation (2), we successfully recovered the entire exponent k.
– When the relation between two partial exponents $k^{(0)}$ and $k^{(1)}$ is far from equation (2), one of the partial exponents $k^{(0)}$ or $k^{(1)}$ can be recovered.

Countermeasure. For an implementation of our parallel algorithm for modular exponentiation, possible countermeasures could be similar way as the case of traditional implementation of modular exponentiation with Montgomery's method. Always performing ("*dummy*") extra subtraction strategy could be a counter measure [HQ00,Wa99], but lead extra computational cost.

6 Conclusion and Further Work

In this paper we described an algorithm for parallel modular exponentiation, and then the computational efficiency has been discussed. Moreover, we showed a timing attack against an implementation of our parallel algorithm.

If one of the partial exponent $k^{(0)}$ or $k^{(1)}$ is recovered, that is half bits of given exponent k is recovered, some number theoretic approaches (e.g. [BDF98]) could be applicable for total break.

Moreover, we demonstrated that the similar strategy as Dhem's attack can be applicable to our parallel modular exponentiation. But we showed the difficulty of the timing attack, which is caused from the parallelism.

To extend our attack to larger n and k, possible further work could be: Attacking on the modular square phase may be useful for the timing attack [DKLMQW98]. Error detection strategies [DKLMQW98,Sc00,SQK01] for the parallel exponentiation should be considered.

References

[BDF98] D. Boneh, G. Durfee, and Y. Frankel, "An attack on RSA given a small fraction of the private key bits," *Advances in Cryptology – ASIACRYPT'98*, LNCS, **1514** (1998), Springer-Verlag, 25–34.

[DKLMQW98] J.F. Dhem, F. Koeune, P.A. Leroux, P. Mestré, and J.J. Quisquater, "A practical implementation of the timing attack," *CARDIS 1998*, LNCS, **1820** (1998), Springer-Verlag, 175–190.

[GG02] J.M.G. Garcia and R.M. Garcia, "Parallel algorithm for multiplication on elliptic curves," *Cryptology ePrint Archive*, Report **2002/179**, (2002), `http:eprint.iacr.org`.

[HQ00] G. Hachez and J.J. Quisquater, "Montgomery exponentiation with no final subtractions: Improved Results," *Cryptographic Hardware and Embedded Systems – CHES 2000*, LNCS, **1965** (2000), Springer-Verlag, 293–301.

[IT02] T. Izu and T. Takagi, "Fast parallel elliptic curve multiplications resistant to side channel attacks," *Public Key Cryptography – PKC 2002*, LNCS, **2274** (2002), Springer-Verlag, 335–345.

[IYTT02] K. Itoh, J. Yajima, M. Takenaka, and N. Torii, "DPA countermeasures by improving the window method," *Cryptographic Hardware and Embedded Systems – CHES 2002*, (2002).

[KJJ99] P.C. Kocher, J. Jaffe, and B. Job, "Differential power analysis," *Advances in Cryptology – CRYPTO'99*, LNCS, **1666** (1999), Springer-Verlag, 388–397.

[Ko96] P.C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," *Advances in Cryptology – CRYPTO'96*, LNCS, **1109** (1996), Springer-Verlag, 104–113.

[Mo85] P.L. Montgomery, "Modular multiplication without trial division," *Math. Comp.*, **44** (no. 170) (1885), 519–521.

[OS00] K. Okeya and K. Sakurai, "Power analysis breaks elliptic curve cryptosystems even secure against the timing attack," *INDOCRYPT 2000*, LNCS, **1977** (2000), Springer-Verlag.

[Sc00] W. Schindler, "A timing attack against RSA with the Chinese Remainder Theorem," *Cryptographic Hardware and Embedded Systems – CHES 2000*, LNCS, **1965** (2000), Springer-Verlag, 109–124.

[SQK01] W. Schindler, J.-J. Quisquater, and F. Koeune, "Improving divide and conquer attacks against cryptosystems by better error detection correction strategies," *Proc. of 8th IMA International Conference on Cryptography and Coding*, (2001), 245–267.

[Wa99] C.D. Walter, "Montgomery exponentiation needs no final subtractions," *Electric Letters*, LNCS, **35** (no. 21) (1999), 1831–1832.

[WT01] C.D. Walter and S. Thompson, "Distinguishing exponent digits by observing modular subtractions," *RSA Conference 2001*, LNCS, **2020** (2001), Springer-Verlag, 192–207.

A Fast Correlation Attack for LFSR-Based Stream Ciphers

Sarbani Palit[1], Bimal K. Roy[2], and Arindom De[3]

[1] Computer Vision & Pattern Recognition Unit
sarbanip@isical.ac.in
[2] Applied Statistics Unit
bimal@isical.ac.in
[3] Computer Vision & Pattern Recognition Unit
arindom_de@hotmail.com

Indian Statistical Institute, INDIA

Abstract. This paper describes a novel fast correlation attack of stream ciphers. The salient feature of the algorithm is the absence of any pre-processing or iterative phase, an usual feature of existing fast correlation attacks. The algorithm attempts to identify a number of bits of the original linear feedback shift register (LFSR) output from the received bits of the ciphertext. These are then used to construct a system of linear equations which are subsequently solved to obtain the initial conditions. The algorithm is found to perform well for LFSRs of large sizes but having sparse polynomials. It may be noted that such polynomials have low Hamming weight which is one more than the number of feedback connections or "taps" of the corresponding LFSR. Its performance is good in situations even where limited cipherlength is available. Another important contribution of the paper is a modification of the approach when the LFSR outputs are combined by a function which is correlation immune and perhaps, unknown to the decrypter.

Keywords: Stream cipher, Correlation attack, LFSR polynomial, Correlation immune function.

1 Introduction

Stream ciphers form an important class of cipher systems. Their speed over that of block ciphers and less complex hardware circuitry make it advantageous to use stream ciphers in many applications [2]. Such ciphersystems are widely used in military applications across the world [3].

In a binary additive stream cipher, the ciphertext is produced by bit-wise addition of the plaintext with the key-stream, all in binary. The key-stream generator is initialized using a secret key. A popular key-stream generator used in stream ciphers consists of several LFSRs combined through a non-linear boolean function. LFSRs lie at the heart of most key-stream generators since they produce pseudo-random sequences with good statistical properties and are easily

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 331–342, 2003.

implemented in hardware. The secret key, unknown to the decrypter, is normally chosen to be the initial conditions of the LFSRs. The LFSR polynomials are assumed to be known.

The objective of the nonlinear combining function is to destroy the inherent linearity present in LFSR sequences. It enables the key-stream to have a large linear complexity in order to prevent linear cryptanalysis. However, depending on the order of resiliency of the function, there is still some correlation between the ciphertext and the LFSR outputs. Attacks that exploit the similarity between the ciphertext and the LFSR outputs, are termed correlation attacks. The nature of the cipher system allows each LFSR to be analysed separately, thus leading to a divide-and-conquer strategy. Such attacks were first proposed in [14] and further developed in [17,4,6,7,11]. The idea of a fast correlation attack, which eliminates the need for an exhaustive search of the LFSR initial conditions was first proposed by Meier and Staffelbach [10]. The algorithm was designed to work for a small number of taps. A number of fast correlation attacks were later proposed in [8,9,5,2,12].

It is important to note, however, that the existing fast correlation attacks suffer from one or more of the following drawbacks:

1. The presence of a preprocessing phase of considerable complexity which naturally increases the overall decoding time.
2. An iterative phase which takes time to converge.
3. The assumption of a combining function that is not correlation immune [15] and also known to the decrypter.

The algorithm proposed in this paper is free of all these restrictions. It may be mentioned in this context, that a correlation attack for a situation where the combining function is both correlation immune and unknown was first proposed in [13]. Recovery of the unknown combining function has also been addressed in [3].

Since a divide-and-conquer approach is possible, each LFSR can be analysed independently of the others. In this situation, it is reasonable to model the ciphertext as the output of a binary symmetric channel through which the output of the ith LFSR passes. The error probability of the channel (or the probability that a bit of the ciphertext is not equal to the corresponding bit of the LFSR sequence) is denoted as $(1-p)$. The proposed approach is based on the fact that every bit of an LFSR equation satisfies certain relations called parity checks, all of which are derived from its generating polynomial [10]. Since the cipherstream is correlated to the LFSR output (with probability of concurrence, p, different from .5), many of its bits will also satisfy the same equations. The extent to which a bit satisfies these equations will determine whether it is actually equal to the LFSR bit at the same position. Once a sufficient number of bits have been correctly determined (slightly more than the length of the LFSR), the initial conditions of the corresponding LFSR are obtained by solving linear equations. Section 2 develops the background for the proposed method while the actual algorithm is laid down in Section 3. Performance of the algorithm, comparison

with other well-known algorithms [10,5] are presented in Section 4. Modifications of the algorithm for decryption involving an unknown combining function which may be correlation immune is discussed in Section 5.

2 The Probabilistic Background

The relationships developed in this section (which are used in the present work) have been worked out in much greater detail in [10]. We merely present them here for the sake of clarity and completeness.

Let C denote the ciphertext, N its length available, X_i the sequence produced by the ith LFSR, d_i and t_i the size and number of feedback taps, respectively, of the ith LFSR. With $P(A)$ standing for the probability of the occurrence of the event A, it follows that $p = P(C = X_i)$[1]. Without loss of generality, since only one LFSR will be considered at a time, let $X_i = X$, $d_i = d$ and $t_i = t$. Further, let the LFSR polynomial (assumed to be known) used to generate X, be represented by

$$a(Y) = 1 + a_1 Y + a_2 Y^2 + \cdots + a_d Y^d \tag{1}$$

The number of non-zero coefficients a_i give the number of taps t. Since the sequence $X = \{x_1, x_2, \cdots\}$ is generated from $a(Y)$

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \cdots + a_d x_{n-d} \tag{2}$$

holds with t number of terms on the right hand side. For every bit of X (except those towards the beginning and end of the sequence), placing it in one of the $t + 1$ positions of (2) yields $t + 1$ relations or equations. Now, the number of relations satisfied by each bit can be further increased by noting that every polynomial multiple of $a(Y)$ gives a linear relation satisfied by X. This is true, in particular for powers $a(X)^j$ with $j = 2^i$. It is also important to note that the resulting polynomials all have t feedback taps which is a basic requirement of the algorithm. Hence, repeated squaring (the number of times limited by the cipherlength available) of the LFSR polynomial followed by shifting each bit from one of $t + 1$ positions to another results in a large number of relations all satisfied by the LFSR sequence X.

Let m be the number of relations satisfied by a particular bit x_n (It has been shown in [10] that the average number of relations satisfied per bit is approximately $(t+1)\log_2(N/2d)$). These relations may be expressed in the form:

$$L_i = x_n + w_i = 0 \qquad i = 1, \cdots, m \tag{3}$$

where w_i represents a sum of exactly t different remaining terms with x_n in one of the $t + 1$ positions in (2) and also its multiples.

Consider now a bit of the cipherstream, c_n in place of x_n in (3) with

$$L_i = c_n + z_i \qquad i = 1, \cdots, m \tag{4}$$

[1] Here, the event $C = X_i$ means that a particular bit of C is the same as the corresponding bit of X_i

with z_i representing a sum of exactly t different remaining terms with c_n in one of the $t+1$ positions in (2) and its multiples.

In this case, L_i may not be equal to zero.

Now, let $w_i = w_{i1} + w_{i2} + \cdots + w_{it}$ and $z_i = z_{i1} + z_{i2} + \cdots + z_{it}$ where, w_{ij} and z_{ij}, $j = 1, \cdots, m$ are binary variables, all independent and identically distributed with equal probability of being 0 or 1. Note that $P(x_n = c_n) = p = P(w_{ij} = z_{ij})$.

Then, $s(t) = P(w_i = z_i)$ can be recursively computed as follows:

$$\begin{aligned} s(1) &= p \\ s(j) &= ps(j-1) + (1-p)(1-s(j-1)) \qquad j = 2, \cdots, t \end{aligned} \tag{5}$$

Observe that, for a particular ciphertext bit to satisfy the ith relation *i.e.* $L_i = c_n + z_i = 0$, either $c_n = x_n$ **and** $w_i = z_i$ or $c_n \neq x_n$ **and** $w_i \neq z_i$. Hence

$$\begin{aligned} P(L_1 = \cdots = L_h = 0; L_{h+1} = \cdots = L_m = 1) = ps^h(1-s)^{m-h} \\ + (1-p)(1-s)^h s^{m-h} \end{aligned}$$

where $s = s(t)$.

Let

$$R = P(c_n = x_n;\ c_n \text{ satisfies at least } h \text{ of } m \text{ relations}),$$

$$Q = P(c_n \text{ satisfies at least } h \text{ out of } m \text{ relations}),$$

$$T = P(c_n = x_n; \text{given that } c_n \text{ satisfies at least } h \text{ of } m \text{ relations}).$$

Then, using (6)

$$R = \sum_{i=h}^{m} \binom{m}{i} ps^i(1-s)^{m-i} \tag{6}$$

$$Q = \sum_{i=h}^{m} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}) \tag{7}$$

$$T = R/Q \tag{8}$$

The quantity h/m which is the minimum fraction of equations that a bit of the cipherstream must satisfy, shall be henceforth, referred to as the upper threshold.

Further, let

$$W = P(c_n \neq x_n;\ c_n \text{ satisfies at most } h \text{ of } m \text{ relations}),$$

$$D = P(c_n \text{ satisfies at most } h \text{ out of } m \text{ relations}),$$

$$E = P(c_n \neq x_n;\ c_n \text{ satisfies at most h of m relations})$$

Then, using (6)

$$W = \sum_{i=0}^{h} \binom{m}{i} (1-p)s^i(1-s)^{m-i} \tag{9}$$

$$D = \sum_{i=0}^{h} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}) \tag{10}$$

$$E = W/D \tag{11}$$

The maximum fraction of equations h/m, that a bit can satisfy in order to be designated as wrong shall be called the lower threshold. Note that the value of h for the upper threshold is different from that of h for the lower threshold.

3 Development of the Algorithm

In order to understand the relationship between the various quantities involved in (8),(7),(11),(10), the expressions have been computed for $d = 31$, $k = 2$, $N = 14,000$ and $p = 0.64$. Figure 1a shows the plot of T vs. the upper threshold (varying h/m). Figure 1b shows the plot of the number of bits determined correctly with probability T out of those satisfying the upper threshold (which is equal to $T*Q*N$) vs. the upper threshold Note that satisfying the upper threshold h/m implies satisfying at least h out of m relations. Figure 1c shows the plot of E vs. the lower threshold. Figure 1d shows the plot of the number of bits determined wrongly (and hence to be complemented), vs. the lower threshold.

It can be seen from Figures 1a and 1b, that as the upper threshold is increased, the probability of correctly determining the bits increases while the number of bits correctly determined decreases. The reverse situation occurs in Figures 1c and 1d *i.e.* as the lower threshold is increased, the probability that a bit is wrong decreases while the number of wrong bits increases. It can hence be concluded that the thresholds must be chosen with a trade-off in mind. Not only must the choice ensure a desired probability of correct determination of the bits but at the same time, make it possible that a required number of these are obtained.

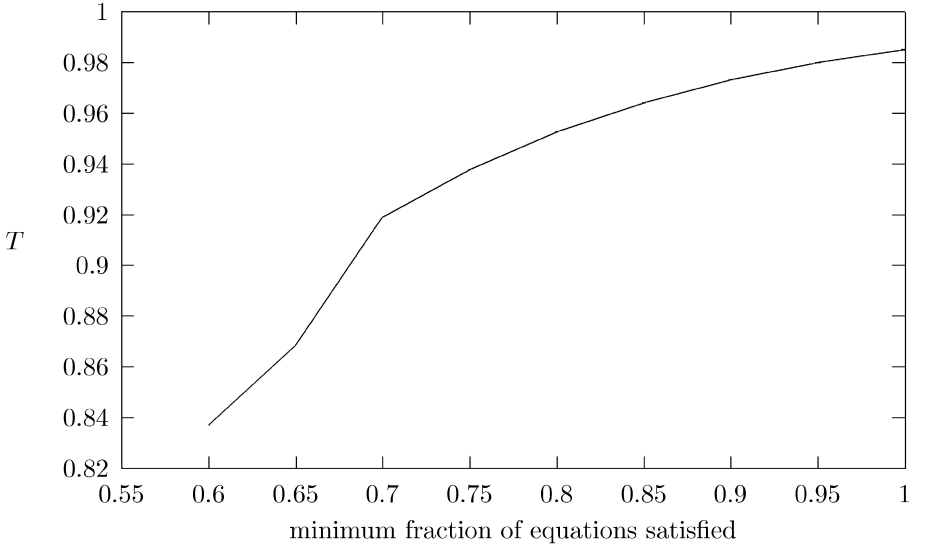

Fig. 1a. T vs. upper threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

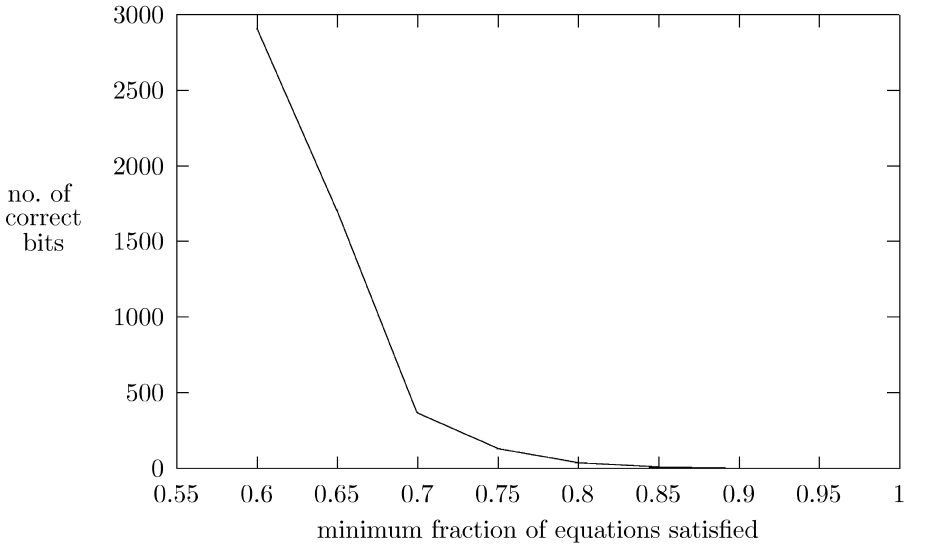

Fig. 1b. No. of correct bits vs. upper threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

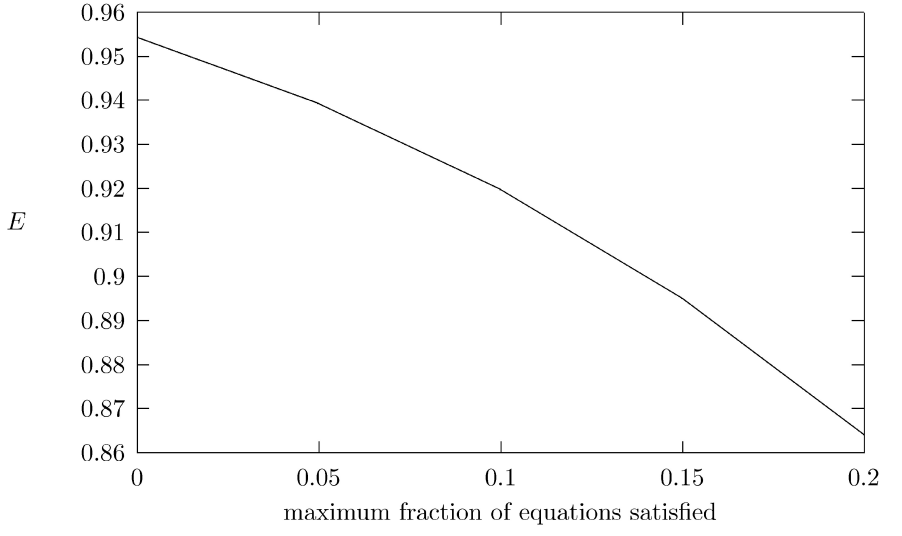

Fig. 1c. E vs. lower threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

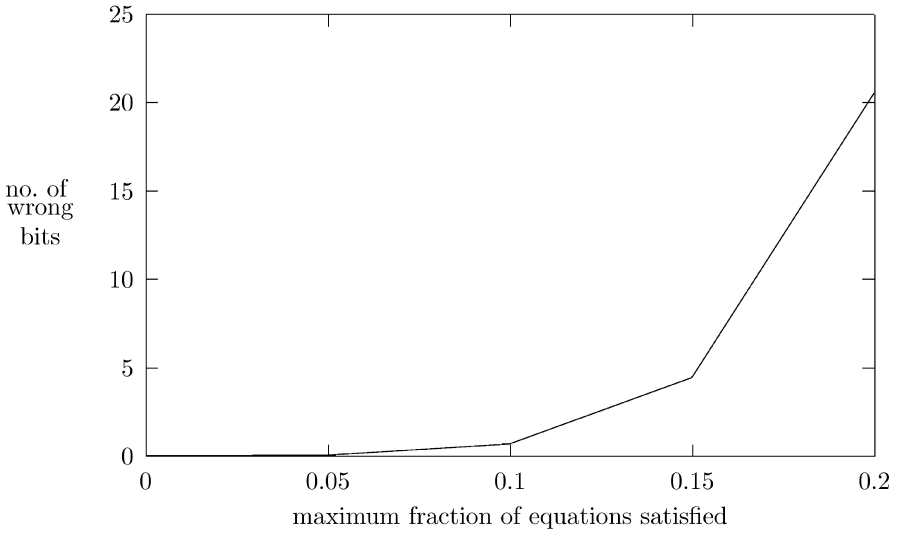

Fig. 1d. No. of wrong bits vs. lower threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

The algorithm can now be outlined as follows:

1. For every bit of the cipherstream, generate as many equations as possible by shifting, squaring etc. the original LFSR feedback polynomial. Compute the percentage of relations, say r, satisfied by each bit.
2. Using (8) and (7), obtain the upper threshold such that the probability of correct determination is at least 0.95 and the number of bits correctly determined is at least equal to d, the length of the LFSR.
3. Using (11) and (10), obtain the lower threshold such that the probability of wrongful determination is at least 0.95. Complement these bits.
4. Express the bits thus determined in terms of the initial conditions of the LFSR and solve the resultant linear system in order to recover the initial conditions. Do this for all combinations of the identified bits, taken d at a time. Note that the system may not always be solvable in which case that particular combination of bits must be rejected.
5. From a plot of the frequency distribution of the occurrence of the initial conditions determined in the above step, locate the set/sets occurring most frequently. When this yields only one set, it can be safely assumed that this is the correct initial condition. If there are multiple sets, the Hamming distance between the LFSR output corresponding to each of these and the cipherstream must be computed. The set with the lowest Hamming distance will be the desired initial condition.

4 Performance

Experiments were conducted on LFSRs of various sizes, having up to 6 taps. In each case, the initial conditions were recovered correctly. An illustration of step 4 of the algorithm can be seen in Table 1. It shows the variation of the Hamming distances (normalized by the cipherlength) for the LFSR outputs corresponding to ten candidate sets of initial conditions which occurred most frequently upon solving the linear system of equations as in step 3 for a length 31, 2 tap LFSR with $p = 0.69$. Index no.1 corresponds to the correct initial condition and it clearly has a much lesser Hamming distance than the others.

Table 1. Normalized Hamming distance for various candidate initial conditions.

index	Normalized Hamming distance
1	0.31
2	0.49
3	0.5
4	0.51
5	0.5
6	0.49
7	0.49
8	0.5
9	0.48
10	0.51

It should also be mentioned that the value m used in the theoretical calculations is only an average number and the number of relations for the bits in the central part of the cipherstream is much more. This explains the improved performance of the algorithm in practice. For example, for the 31 length, 2 tap LFSR with $p = 0.69$ considered here, only 4000 cipherbits were sufficient for recovering 36 bits correctly while from theoretical calculations, 14,000 cipherbits appeared to be required. As observed from the graphs of Section 3, the number of bits determined correctly depends on the value selected for the upper threshold, increasing for lower values of the threshold. It actually decreases for higher values of the upper threshold.

Figure 2 shows a plot of $p = P(C = X_i)$ vs. the number of taps for a length 100 LFSR and $N = 20,000$. The value of p is determined using (8) and (7) so that at least 100 bits are correctly determined with a probability greater than 0.95 . Since the contribution from complementing the incorrect bits in this case turned out to be very small, it has been ignored here. As expected, the required correlation between the cipherstream and LFSR sequence increases with the number of taps.

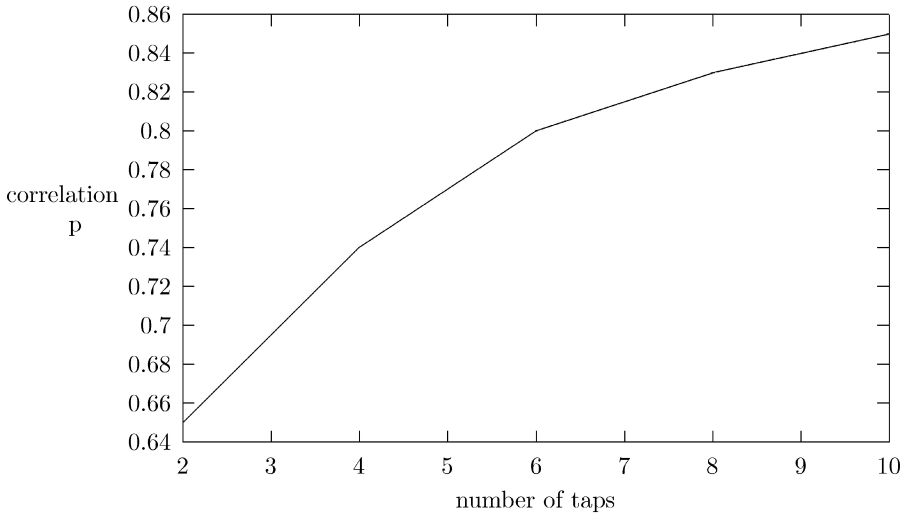

Fig. 2. $p = P(C = X_i)$ vs. no. of taps for $d = 100$, $N = 20000$

From the algorithm, it is obvious that the computational time largely depends on step 4 and consequently, on the number of bits recovered with a certain probability of being correct. Let this number be $d + \Delta$. Then, the maximum number of times that d linear equations have to be solved for the d initial conditions is

$$N_l = \binom{d + \Delta}{d} \tag{12}$$

The choice of Δ thus decides the computation time. A reasonable approach is to go through step 4 with a small value of Δ, say 5% of d. However, if this does not yield a clear solution for the unknown initial conditions, the step must be repeated using a higher value for Δ. It has been observed that higher values are necessary as the number of taps increases. It is also to be noted that a judicious choice of the upper threshold yields a smaller number of bits with a greater chance of being correct, both of which act in the interest of reducing the computational time.

In order to compare the performance of our method with some standard algorithms, some case studies were made. Algorithm B (henceforth referred to as Alg. B) of [10] and Algorithm A1 (henceforth referred to as Alg. A1) of [5] were used for comparison. For the LFSR of length 31, with 2 taps, and $p = 0.69, 0.71, 0.75$, the proposed algorithm and Alg. A1 successfully recover the correct initial condition in each case with a cipherlength of 4000 bits. It was noted, however, that the time taken by Alg. A1 is more than double that of the time taken by the proposed algorithm. Alg. B failed to determine the correct LFSR sequence in each case.

With an LFSR of length 31, 4 taps, $p = 0.75$ and message length of 4000, both the proposed method and Alg. B are successful. Alg. A1 however, fails even with a cipherlength of 10000.

5 Unknown/Correlation-Immune Combining Function

When the combining function is unknown, so is the value of p. Hence the selection of the thresholds, based on the values of T, Q, E and D is no longer directly possible. In such a situation, a search over all possible initial conditions, for the one which has the maximum number of coincidences with the ciphertext, as suggested in [13], works well. Once the initial conditions have been determined, the combining function can be estimated as proposed in [13]. However, the attack can no longer be said to be a fast correlation one since it involves examining all the initial conditions. In such a situation, knowledge of the degree of correlation immunity of the combining function can be appropriately employed.

It has been shown in [16,1] that a Boolean function f is mth-order correlation immune if and only if it is not correlated to linear functions of any subset of m input variables. However, f may be correlated to a linear function of $m+1$ variables, in which case, a correlation attack may be mounted on it. Similarly, in an LFSR based stream cipher system, even though the combining function may not be correlated to the individual LFSR outputs, it may be correlated to certain combinations of them. Hence, it is necessary to determine these LFSR combinations. It is known that a combination of LFSRs (whose outputs are obtained by EX-ORing the individual outputs) may be equivalently replaced by an LFSR whose feedback polynomial is given by the product of constituent LFSR feedback polynomials. For example, if $a(X_i)$ is the generating polynomial of X_i, then $(a(X_1))(a(X_2))$ is the generating polynomial of $(X_1 \oplus X_2)$. If it is known that the combining function is correlated to the output of a particular combination of LFSRs, then it will be correlated to the output of the equivalent LFSR. The algorithm of Section 3 can now be used to determine this output. When the function is unknown, so is the knowledge of such combinations and they must be determined systematically. For instance, if the function is correlation immune of order 1, a combination of two of the LFSRs must be considered. Each of the correctly identified bits is expressed in terms of the initial conditions of either LFSR and hence, both the sets of initial conditions are recovered simultaneously. In order to test this approach, a combining function

$$f = X_3 \oplus X_4 \oplus X_1 X_2$$

was used to combine the LFSRs X_1, X_2, X_3 and X_4. The polynomials used were $(x^{10}+x^3+1)$, (x^9+x^4+1), (x^7+x+1) and (x^6+x^5+1), respectively. Note that $P(f = X_i; i = 1, \cdots, 4) = 0.5$ but $P(f = (X_3 \oplus X_4)) = 0.75$. Hence, the algorithm developed in Section 3 is employed with the corresponding LFSR polynomial $((x^7+x+1) \times (x^6+x^5+1)$. With a cipherlength of 8000, both the sets of initial conditions were recovered successfully.

6 Conclusions

The method presented in this paper appears to perform better than Alg. B of [10] especially when smaller cipherlengths are available and the number of taps is small. At the same time, it eliminates the need for further iterations,

which is an essential feature of Alg. B. It also compares favourably with Alg. A1, being faster and also requiring smaller cipherlength. Our approach does not have a computationally intensive and time consuming pre-processing stage as with other contemporary fast correlation attack algorithms such as Alg. A1. For example, for a polynomial of length 60 and 3 taps and a cipherlength of 14,20,000, the authors of Alg. A1 report a precomputation time of about four days. An important step of the proposed algorithm consists of successively solving a set of linear equation. This set may be made as small as desired, limited only by the length of the LFSR. It may be said therefore that the proposed algorithm requires neither pre-computation nor iterative convergence.

As suggested in the paper, correlation attacks are indeed possible even for unknown combining functions which are correlation immune of a particular degree, provided that sufficient computational power is available. However, the restriction on the number of taps is a bottleneck of such attacks. For the attack on a system with a function which is correlation immune of some order, it should be noted that the generating polynomial of the equivalent LFSR (which is equivalent to the combination of LFSRs, as explained in Section 5) with whose output the cipherlength is correlated, may have a greater number of taps than the constituent LFSRs. However, this does not pose a problem as long as the length of the equivalent LFSR is large enough. In fact, even values of p close to 0.5 may be handled if the LFSR has a very long length and a sufficient amount of ciphertext is available. It may be emphasized again that our observations strengthen the existing views that in order to prevent correlation attacks, the designer needs to choose feedback polynomials with a large number of taps, have a large number of LFSRs and a combining function with a sufficiently high degree of correlation immunity.

References

1. L. Brynielsson, "A short proof of the Xiao-Massey lemma," *IEEE Transactions on Information theory*, vol. IT-35, no.6 (1989), p. 1344.
2. A. Canteaut and M. Trabbia,"Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5," *EUROCRYPT 2000* , LNCS 1807, B. Preneel Ed., Springer-Verlag, Berlin Heidelberg 2000, pp. 573–588.
3. A. Canteaut and E. Filiol, "Ciphertext only reconstruction of stream ciphers based on combination generators," *Fast Software Encryption – FSE 2000*, LNCS 1978, Springer Verlag 2001, pp.165–180.
4. V. Chepyzhov and B. Smeets, "On a fast correlation attack on stream ciphers," *Advances in Cryptology – EUROCRYPT '91*, Vol. 547, D.W. Davies, editor, Springer Verlag, 1991, pp. 176–185.
5. V.V. Chepyzhov, T. Johansson and B. Smeets, "A simple algorithm for fast correlation attacks on stream ciphers," *Fast Software Encryption*, 2000.
6. A. Clark, J. Golić and E. Dawson, "A comparison of fast correlation attacks," *Fast Software Encryption*, Third International Workshop (LNCS 1039), D. Gollman, editor, Springer Verlag, 1996, pp. 145–157.
7. R. Forre, "A fast correlation attack on non-linearly feedforward filtered shift register sequences," *Advances in Cryptology – EUROCRYPT '89* (LNCS 434), 1990, pp. 586–95.

8. T. Johansson and F. Jönsson, "Improved fast correlation attacks on stream ciphers via convolutional codes," *Proceedings of Cryptology – EUROCRYPT '99*, Springer Verlag, LNCS 1592, pp. 347–362.
9. T. Johansson and F. Jönsson, "Fast correlation attacks based on Turbo Code techniques," *Proceedings of Cryptology – Crypto '99*, Springer Verlag, LNCS 1666, pp. 181–197.
10. W. Meier and O. Staffelbach,"Fast correlation attacks on certain stream ciphers," *Journal of Cryptology*, vol.1, no.3, 1989, pp. 159–176.
11. M.J. Mihaljevic and J. Golić, "A comparison of cryptanalytic principles based on iterative error-correction," *Advances in Cryptology – EUROCRYPT '91*, Vol. 547, D.W. Davies, editor, Springer Verlag 1991, pp. 527–531.
12. M. Mihaljević, M. P. C. Fossorier and H. Imai, "Fast Correlation Attack Algorithm with List Decoding and an Application," *Fast Software Encryption- FSE 2000.*
13. S. Palit and B.K. Roy, "Cryptanalysis of LFSR-encrypted codes with unknown combining function," *Advances in Cryptology – ASIACRYPT '99*, Vol. 1716, Springer 1999, pp. 306–320.
14. T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Transactions on Computers*, Vol. c-34, No.1, January 1985, pp. 81–85.
15. T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining functions for Cryptographic Applications," *IEEE Transactions on Information Theory*, Vol. 30, No. 5, September 1984, pp.776–780.
16. G. Xiao and J.L. Massey, " A spectral characterization of correlation-immune combining functions," *IEEE Transactions on Information theory*, vol. IT-34, no.3 (1988), pp. 564–571.
17. K. Zeng and M. Huang, "On the linear syndrome method in cryptanalysis," *Advances in Cryptology – CRYPTO '88*, Vol. 403, S. Goldwasser, editor, Springer Verlag 1990, pp. 469–478.

Making the Key Agreement Protocol in Mobile Ad Hoc Network More Efficient

Gang Yao[1,2], Kui Ren[1], Feng Bao[1], Robert H. Deng[1], and Dengguo Feng[2]

[1] InfoComm Security Department, Institute of InfoComm Research,
21 Heng Mui Keng Terrace, Singapore 119613
{yaogang, renkui, baofeng, deng}@i2r.a-star.edu.sg
[2] The State Key Laboratory Of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, P.R.China
fengdg@263.net

Abstract. Mobile ad hoc networks offer convenient infrastructureless communications over the shared wireless channel. However, the nature of mobile ad hoc networks makes them vulnerable to security attacks, such as passive eavesdropping over the wireless channel and denial of service attacks by malicious nodes. To ensure the security, several cryptography protocols are implemented. Due to the resource scarcity in mobile ad hoc networks, the protocols must be communication efficient and need as less computational power as possible. Broadcast communication is an important operation for many application in mobile ad hoc networks. To securely broadcast a message, all the members in the network need share a group key so that they can use efficient symmetric encryption, such as DES and AES. Several group key management protocols have been proposed. However, not all of them are communication efficient when applied to mobile ad hoc networks. In this paper, we propose a hierarchical key agreement protocol that is communication efficient to mobile ad hoc networks. We also show how to manage the group efficiently in a mobile environment.

Keyword: Mobile ad hoc networks, key agreement, hierarchical, security.

1 Introduction

An ad hoc network is a collection of nodes dynamically forming a peer to peer network without the use of any existing network infrastructure or centralized administration. Mobile ad hoc networks have attracted significant attentions recently due to its wide applications in different areas. Nodes of a mobile ad hoc network are often mobile, and the network can be formed, merged together or partitioned into separate network on the fly, without necessary relying on a fix infrastructure to manage the operation. Mobile nodes that are within the communication range of each other can communicate directly, whereas the nodes that are far apart have to rely on intermediary nodes (routers) to relay messages. Mobile ad hoc networks can be used for emergency, law enforcement and rescue

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 343–356, 2003.

missions. Since, the cost to set up a mobile ad hoc network is low, it is also a very attractive option for commercial uses.

Security is one of the most important implementation issues for mobile ad hoc networks and it must be solved. There are four fundamental security issues which must be addressed: confidentiality, authentication, integrity and availability. Because of the high level of self-organization, dynamic topology, dynamic membership or vulnerable wireless link, mobile ad hoc networks are difficult to secure. In addition, security solutions applied in more traditional networks may not directly be suitable for protecting them.

Because the mobility of a node in the mobile ad hoc network causes frequently changed network topology, any security solution with a static configuration is not suitable. Till now, almost all cryptography protocols are based on secret keys or public keys. Public key based protocols have some inherent advantages over the secret key algorithms. However, it is well-known that the secret key based encryption algorithms (such as DES, AES) is much faster than the public key based protocols (such as RSA, ElGamal). In this paper, we will concentrate on how to build a common secret key for a group so that they can communicate securely. Several group key management protocols ([1], [2], [11], [12], [17], [18]) have been proposed for wired networks. However, not all of them are communication efficient when applied to mobile ad hoc networks. In this paper, we propose a hierarchical key agreement protocol that is communication efficient by using hierarchical networks to set up clusters among a group of mobile nodes. We apply some existed group key agreement protocols for each cluster at each level. As long as the selected protocols used in our protocol are secure, our protocol is also secure unambiguously.

The rest of the paper is organized as follows. In section 2, we recall some basic definitions and review several previous key agreement protocols. In section 3, we present our hierarchical key agreement protocol and give the communication cost, lasting time and total exponentiation of the protocol. In section 4, we show how to dynamic maintain the members in our protocol. And the conclusion is given in section 5.

2 Preliminaries

In this section, we present some definitions and general terminologies used in this paper, and we also review some related work.

2.1 Basic Definitions

The definitions have been adapted from [2], [18] and [15]. In general, the key establishment protocols can be classified into two types: key distribution protocols and key agreement protocols. The key distribution protocols, sometimes called as *centralized* key distribution protocols, are generally based on a trusted third party (TTP). The key agreement protocols, on the other hand, do not use a TTP but rely on the group members for a general key agreement. A *key agreement*

protocol is a key establishment method in which, a shared secret key is derived by two or more specified parties as a function of information contributed by, or associated with, each of these, such that no party can predetermine the resulting value.

Let $\mathcal{P}$ be a n-party key agreement protocol, and $\mathcal{M}$ be the set of members in the protocol and let S_n be a secret key generated as a result of $\mathcal{P}$. The protocol $\mathcal{P}$ is said to provide *implicit key authentication* if each $M_i \in \mathcal{M}$ is assured that no party $Q \notin \mathcal{M}$ can learn the key S_n. Then this protocol is called an *authenticated key agreement*. The protocol $\mathcal{P}$ provides *key confirmation* if any member is assured that its peer(s) do in fact, possess the particular key S_n. A key agreement protocol offers *perfect forward secrecy* if the compromising of a long-term key S_n cannot result in the compromising of the keys generated before S_n. On the other hand, a key agreement protocol is said to be vulnerable to *known key attacks* if the compromising of past keys allows a passive adversary to get future group keys, or an active adversary to impersonate one of the protocol members. For more detailed discussions of the above definitions, see [2], [4] and [5].

Then we present some of the definitions used in a mobile ad hoc environment. Assume that all mobile nodes are given as a set V of n nodes. Each node is assumed to have some computational power and an omni-directional antenna. A message sent by a node can be received by all nodes within its transmission range. Here we assume every node has the same maximum transmission range which is normalized to one unit. All these nodes induce a *unit disk graph* UDG(V), in which, there is an edge between two nodes if and only if the distance between them is at most one unit. The UDG(V) is always assumed to be a connected graph. All the nodes within a constant k-hop neighborhood of a node $u \in V$ are the k-local nodes or k-hop neighbors of u, represented by $N_k(u)$ hereafter. All nodes are assumed to be almost static for a reasonable period of time.

A type of hierarchical ad hoc network structure is called *an ad hoc network with mobile backbones* (MBN). In the mobile ad hoc networks, mobile nodes are first dynamically grouped into 1-hop clusters. Each cluster elects a cluster head to be a *backbone node* (BN). Among the mobile nodes, backbone nodes have an additional powerful radio to establish wireless links among themselves. Thus, higher level links are established to connect the BNs into a network, and we call this higher level network a *backbone network*. Since the backbone nodes are also moving and join or leave the backbone network dynamically, the backbone network is exactly an ad hoc network running in a different radio level. Multilevel MBNs can be formed recursively in the same way. For more detailed discussions of the above definitions, see [10], [16] and [21].

2.2 Some Secure Group Key Agreements

Before we review the group key agreement protocols, we describe the notations used in these protocols:

n	number of members in the protocol
i, j, k	indices of members (range $[1, n]$)
M_i	i-th group member
q	order of an algebraic group G
α	exponential base delimited by q
r_i	random exponent generated by M_i
K	Group key shared among n members

The Diffie-Hellman algorithm [8] allows the establishment of a cryptographic secret key between two entities by means of data exchange through an insecure communication channel. The algorithm executed between two entities M_1 and M_2 is defined as follows: Member M_1 sends α^{r_1} to M_2 and M_2 sends α^{r_2} to M_1. M_1 computes the key $K = (\alpha^{r_2})^{r_1}$ and vice-versa for M_2. The security of this protocol is based on the assumption of the difficulty of the discrete logarithm arithmetic and the Diffie-Hellman Decision problem.

Several solutions for extending the Diffie-Hellman key exchange to a multiparty key agreement have been proposed, such as the group Diffie-Hellman protocol [17], the hypercube protocol [7], the octopus protocol [7], the Burmester-Desmedt protocol [6], the tree based protocol [6] and etc.

The generic n-party Diffie-Hellman key exchange protocols were developed by Steiner, Tsudik and Waidner [17]. These protocols suite consists of key management protocols for dynamic groups. Two protocols GDH.2 and GDH.3 were presented.

Algorithm 1: The group Diffie-Hellman protocol GDH.2

- Round i $(1 \leq i \leq n-1)$: M_i sends $\alpha^{\prod_{k=1}^{i} r_k}$ and $\alpha^{(\prod_{k=1}^{i} r_k)/r_j}$ $(\forall\, 1 \leq j \leq i)$ to M_{i+1}.
- Round n: M_n sends $\alpha^{(\prod_{k=1}^{n} r_k)/r_i}$ to each M_i.

Each member computes the final key as $K = (\alpha^{(\prod_{k=1}^{n} r_k)/r_i})^{r_i} = \alpha^{(\prod_{k=1}^{n} r_k)}$.

Algorithm 2: The group Diffie-Hellman protocol GDH.3

- Round i $(1 \leq i \leq n-2)$: M_i sends $\alpha^{\prod_{k=1}^{i} r_k}$ to M_{i+1}.
- Round $n-1$: M_{n-1} sends $\alpha^{\prod_{k=1}^{n-1} r_k}$ to each M_i.
- Round n: M_i sends $\alpha^{(\prod_{k=1}^{n-1} r_k)/r_i}$ to M_n.
- Round $n+1$: M_n sends $\alpha^{(\prod_{k=1}^{n} r_k)/r_i}$ to M_i.

Each member computes the final key as $K = (\alpha^{(\prod_{k=1}^{n} r_k)/r_i})^{r_i} = \alpha^{(\prod_{k=1}^{n} r_k)}$.

The hypercube protocol was presented by Becker and Wille [7] as an example of the protocol requiring the minimum number of rounds. Four nodes, which we shall call M_1, M_2, M_3 and M_4, are arranged in a square. They can create a shared key by just four Diffie-Hellman key exchanges:

Algorithm 3: The hypercube protocol $(n = 4)$

- Round 1: M_1 and M_2 exchange keys $(\alpha^{r_1 r_2})$ in the usual way, and M_3 and M_4 exchange keys $(\alpha^{r_3 r_4})$ in the usual way.
- Round 2: Next, M_1 exchanges keys with M_3, using the 2-way key as the secret exponent, and M_2 and M_4 do the same.

The result $K = \alpha^{\alpha^{r_1 r_2} \alpha^{r_3 r_4}}$ is a key that is shared among all four participants. This is the hypercube protocol in the case where the dimension of the cube d equals 2. The 4-way key exchange can be generalized to a protocol for higher numbers of nodes, as long as the number of participants equals 2^d, for $d \in \mathbb{Z}$, and the protocol executes in d rounds.

The octopus protocol is an extension of the hypercube protocol for networks with an arbitrary number of nodes. A subgroup of nodes is arranged in a hypercube, composing a core. Each core node establishes a key with each nearby non-core node using the Diffie-Hellman protocol. The product of these keys is used to establish a key among the core nodes as specified by the hypercube protocol. At last, this key is distributed to the other nodes.

In [6], Burmester and Desmedt presented several group key distribution systems based on public keys. They also extend these protocols to authentication and prove the security provided the Diffie-Hellman problem is intractable. From the ad hoc network point of view, the most interesting protocol is the tree based protocol. This is a key distribution protocol for a network whose topology is in the form of a binary tree. The root is a chair that generates the key and distributes it along the tree. The participants' contributions are only used in encrypting the session key while distributing it.

Algorithm 4: The tree based key distribution protocol

- Round 1: M_i computes $z_i = \alpha^{r_i}$. If $i > 1$, M_i sends z_i to $M_{\lfloor i/2 \rfloor}$; If $2i \leq n$, M_i sends z_i to M_{2i}; If $2i + 1 \leq n$, M_i sends z_i to M_{2i+1}.
- Round 2: M_i computes $K_i = z_{\lfloor i/2 \rfloor}^{r_i}$ if $i > 1$, and $K_{2i+j} = z_{2i+j}^{r_i}$ for $j = 0, 1$, if $2i + j \leq n$.
- Round 3: M_1 selects a session key K, then he sends $Y_{2+j} = K \cdot K_{2+j}$ to M_{2+j} for $j = 0, 1$, and set $l = 0$.
- Round $4+l$: If M_i is at level l of the tree ($\lfloor \log_2 i \rfloor = l$), then M_i decrypts Y_i in order to get K. Next, he sends $Y_{2i+j} = K \cdot K_{2i+j}$ to M_{2i+j} for $j = 0, 1$, if $2i + j \leq n$, and set $l = l + 1$.

Burmester and Desmedt [6] were presented another key agreement protocol which is executed in three rounds. Each participant M_i ($i \in [1, n]$) executes the following operations:

Algorithm 5: The Burmester-Desmedt protocol

- Round 1: M_i generates a secret random value r_i and broadcasts $z_i = \alpha^{r_i}$ to the other participants;
- Round 2: M_i computes and broadcasts $X_i = (\frac{z_{i+1}}{z_{i-1}})^{r_i}$ to the other participants;
- Round 3: M_i computes the group key $K = z_{i-1}^{nr_i} \cdot X_i^{n-1} \cdot x_{i+1}^{n-2} \cdot \ldots \cdot X_{i-2} \mod p$.

This group key has the form $K = \alpha^{r_1 r_2 + r_2 r_3 + \ldots + r_n r_1}$ and shares the security characteristics presented by the Diffie-Hellman algorithm. This protocol is

efficient with respect to the total number of rounds. This characteristic could allow faster execution, but each round requires n simultaneous broadcasts which are usually not possible, even in wireless networks. Another disadvantage is that this protocol makes use of a high number of exponential operations.

At last, we review the hybrid key agreement protocol presented by Li, wang and Frieder [14]. In this protocol, the first round is a clustering method that divides the entire set of the nodes into subgroups based on the geometric locations of the nodes. Each of these subgroups selects a leader (also called dominator). This selection process is done by generating a connected dominating set (CDS) (see [19], [3] and [20]) from the set of wireless nodes. Once the CDS has been constructed, the set of dominators of the CDS form the subgroup leaders; each dominator and the set of its dominatees form an individual subgroup. Then the key agreement protocols such as GDH.2 etc. could be applied to the set of dominators, and the key is generated as a contribution from all the dominators. On success of the key agreement protocol over the dominators, each dominator and the set of its subgroup members follow the key distribution protocol if the same key for each subgroup is required. The afore mentioned steps make sure that all the nodes share the same key, and an overall key agreement is reached.

Algorithm 6: The hybrid key agreement protocol

1. Wireless nodes construct a CDS distributively.
2. The contributory key agreement protocol is applied among the set of computed dominators and connectors.
3. Each dominator distributes the computed key to all its dominatees if the same key is required. Otherwise, each subgroup performs its own key agreement protocol.

Because this protocol just establish one group key and maintain the topology of the network while the node is moving in the network, it is not very efficient for using in the mobile ad hoc network.

3 Hierarchical Key Agreement Protocol

In a mobile ad hoc environment, the number of members could be very large. If all the members participate the process that creates the common secret key, it will be very difficult to manage the process. A method to solve this problem is to build a hierarchical ad hoc network. In each level, all the members are divided into clusters and all the members in a cluster perform a key agreement protocol to get the cluster key. Then all the cluster leaders perform a key agreement protocol to get group key. At last, the group key is distributed to all the group members.

3.1 Protocol

We present a hierarchical key agreement protocol in this subsection. Our protocol is a hierarchical link state based key agreement protocol. When the group key is

being established, the mobile ad hoc network maintains a hierarchical topology, where elected cluster heads at the lowest level become the members of the next higher level. These new members in turn organize themselves in clusters, and so on. The goal of clustering is to establish cluster key with high efficiency. Once the hierarchical structure of the network has been constructed, we could apply the key agreement protocol to establish the group key. On success of the key agreement protocol over each level, each cluster leader and the set of its cluster members follow the key distribution protocol so that the upper level keys can be distributed to the members of this lower level. The afore mentioned steps make sure that all the nodes in the mobile ad hoc network share the same group key, and an overall key agreement is reached. We outline our hierarchical key agreement protocol that is suitable for mobile ad hoc networks as follows:

Algorithm 7: The hierarchical key agreement protocol

- Step 1: Mobile nodes in the ad hoc network construct a hierarchical link networks of h levels. The clusters are independently controlled and are dynamically reconfigured as nodes move.
- Step $i+1$ ($1 \leq i \leq h$): Each cluster in level i:
  1. Each member u random chooses a secret r_u.
  2. All the members agree with a key agreement protocol.
  3. All the members perform the protocol to establish the cluster key.
- Step $h+2$: Each cluster leader distributes the computed upper cluster keys to all the member in the cluster.

For example, suppose that the hierarchical ad hoc network has three levels after all members execute the first step. We denote the cluster in level i by $C_{i,j}$ ($i = 1, 2, 3$), then at the end of step 2, the members in cluster $C_{i,j}$ will share the cluster key $K_{i,j}$. At the third step, each cluster leader distributes the upper cluster keys to all the member in the cluster. In the figure, the group key will be $K_{3,1}$. All the members in cluster $C_{2,1}$ will share the cluster key $K_{2,1}$ and the group key $K_{3,1}$, and all the members in cluster $C_{1,1}$ will share the cluster key $K_{1,1}$, higher cluster key $K_{2,1}$ and the group key $K_{3,1}$. All the members in the mobile ad hoc network can exchange the message encrypted by $K_{3,1}$; the member in the cluster $C_{1,1}$, $C_{1,2}$ and $C_{1,3}$ can exchange the message encrypted by $K_{2,1}$; the member in the cluster $C_{1,1}$ can exchange the message encrypted by $K_{1,1}$.

In addition to multilevel clustering, our protocol also provides multilevel logical partitioning. While clustering in the key establishment phase is based on geographical relationship between the mobile nodes, logical partitioning is based on logical, functional affinity between the mobile nodes. After the group key is established, we will only maintain the logical topology of the clusters, not the geographical topology. When the member in the network moves, if the logical topology of the network does not change, we will not change the group key, although the geographical topology may has been changed.

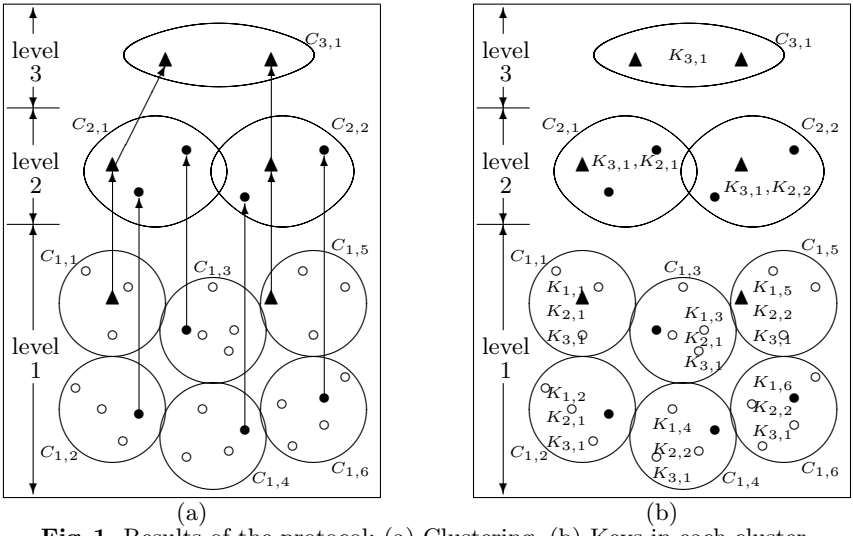

Fig. 1. Results of the protocol: (a) Clustering. (b) Keys in each cluster.

3.2 Analysis

In this subsection, we concentrate on the analysis of the communication complexity, computation complexity and time complexity of our hierarchical key agreement protocol. To simplify the analysis, we assume that the hierarchical ad hoc network has only two levels.

Most key agreement algorithms assume that an existing order is defined in the group members by requiring each member M_i sending a message to M_{i+1}, and the communication cost between any two members M_i and M_{i+1} is always one unit. In practice, these assumptions do not hold in wireless ad hoc environment. The nodes on the backbone are not connected to each other in any specific order, and a direct communication exists only between a node M_i and its 1-hop neighbors. If M_{i+1} is not the 1-hop neighbor of M_i, then the communication from M_i to M_{i+1} must be relayed by other intermediate nodes.

Assume that there are total n wireless nodes. Different clustering algorithms, such as Lowest ID and Highest Degree algorithm (more detail, see [9] and [13]) can be used for the creation of the clusters and the election of cluster header. After the clustering process, there are g clusters $C_1, C_2, \ldots, C_g$ with $n_1, n_2, \ldots, n_g$ members. The members in cluster C_i are denoted by $M_{i,j}$ $(i = 1, 2, \ldots, g, j = 1, 2, \ldots, n_i)$, respectively, and the cluster leader in C_i is denoted by D_i. All the D_i form the backbone network. The communication cost of the clustering step of algorithm 7 is $O(n)$.

In the following discussion, we assume in each cluster, the communication cost by sending a message from one member to the other is one unit, so we can easily get the efficiency parameters for each protocol in any cluster.

	GDH.2	GDH.3	Hypercube	B-D	Tree Based
comm. cost	n_i	$2n_i - 1$	$n_i \log n_i$	$2n_i$	$2n_i - 1$
lasting time	n_i	$n_i + 1$	$\log n_i$	2	$\log n_i + 1$
total exp.	$n_i(n_i + 3)/2 - 1$	$5n_i - 6$	$2n_i \log n_i$	$n_i(n_i + 1)$	$2n_i$

Next, we establish the efficiency parameters for the backbone network. Assume that the average communication cost between two neighbor in the backbone network is c_1 and the broadcasting communication cost in the backbone network is c_2. Assume that the average time for transmitting a message between two neighbor in the backbone network is t_1 and the longest time for transmitting a message between two nodes in the backbone network is t_2. Then, we can easily get the efficiency parameters for each protocol in the backbone network. For the tree based protocol is a key distribution protocol, we do not compare it with the others.

	GDH.2	GDH.3	Hypercube	B-D	Tree Based
comm. cost	$(g-1)c_1 + c_2$	$(g-2)c_1 + 3c_2$	$(g \log g)c_1$	$2gc_2$	$gc_1 + c_2$
lasting time	$(g-1)t_1 + t_2$	$(g-2)t_1 + 3t_2$	$t_1 \log g$	$2t_2$	$t_1 + t_2$
total exp.	$g(g+3)/2 - 1$	$5g - 6$	$2g \log g$	$g(g-1)$	$2g$

It is difficult to establish the minimal communication cost and lasting time in the mobile ad hoc network, for it is dependent on the geographical topology of the network. In the following, we discuss the minimal communication cost and lasting time when the geographical topology of the mobile ad hoc network is stable.

It is easy to see that the total communication cost of this protocol is

$$\sum_i (\text{cost of cluster } C_i) + (\text{cost of backbone}).$$

In our protocol, using GDH.2 in clusters and the backbone network can get the minimal communication cost, and it is n_i in the cluster C_i and $(g-1)c_1 + c_2$ in the backbone network. Thus, the minimal total communication cost of the hierarchical key agreement protocol is $\sum_i n_i + gc_1 + c_2 = n + gc_1 + c_2$.

Because the cluster key agreement can execute paralleled, the total lasting time of the hierarchical key agreement protocol is

$$\max_i \{\text{time of cluster } C_i\} + (\text{time of backbone}).$$

If the broadcasting operation is forbidden, the minimal lasting time can be got by using the hypercube protocol in clusters and the backbone network, and the lasting time of the hierarchical key agreement protocol is $\max_i\{\log n_i\} + t_1 \log g$. If the broadcasting operation can be used, the minimal lasting time can be got by using the Burmester-Desmedt protocol in clusters and the backbone network, and the lasting time of the hierarchical key agreement protocol is $2 + 2c_2$.

The total exponential operations of this protocol is

$$\sum_i (\text{exp. operations of cluster } C_i) + (\text{exp. operations of backbone}).$$

In our protocol, using GDH.3 in clusters and the backbone network can get the minimal total exponential operations, and the total exponential operations of the hierarchical key agreement protocol is $\sum_i (5n_i - 6) + (5g - 6) = 5n - g - 6$.

The last step of the hierarchical key agreement protocol is to distribute the group key to all the group members. In this step, each cluster leader broadcasts a message to all the members in the cluster. Because we assume that all the members in a cluster are in 1-hop area, the total message in each cluster is one. Thus, the communication cost of this step is $O(g)$.

4 Dynamic Maintenance

In the previous protocols, they assumed that the nodes in the network are static or can be viewed static. This is not true in mobile ad hoc networks. Mobile nodes will move around in the network. The movement of the nodes makes it very difficult to design efficient protocols for various applications such as routing, backbone construction, and so on. The hybrid key agreement [14] consider the movement of the nodes, but this protocol maintain the geographical topology of the nodes, and the group key will update when a node moves into or out of a subgroup. If the nodes in the network moves frequency, it will take lots of communication and time to manage the group key, and this makes the hybrid key agreement not very efficient for the mobile ad hoc network. In this section, we study in detail how our hierarchical key agreement protocol can easily manage the group key in the network. If the node's movement does not cause the change of the logical topology of the network, it is obvious that no key updating is necessary, although the geographical topology of the network maybe changes. We detail our hierarchical key agreement how to dynamic manage the group key in the mobile as hoc network. We assume that GDH protocol is used in both the clusters and the backbone network.

4.1 Member Joining

We first show that how a new member u join in the mobile ad hoc network.

Suppose that u find a nearby node D_i which is a leader of cluster C_i, and C_i has n_i members. u sends a message to join the cluster of D_i, and marks itself as a member of cluster C_i. Member addition algorithm [17] is process to get new group key, but this algorithm change the leader of the cluster. For the cluster leader saving same information that used to get group key, it is not a good choice to change the leader of the cluster. We modified that algorithm as follows:

1. We assume that the leader D_i saves the contents of messages he receives.
2. D_i sends $\{\alpha^{\prod\{r_k|k\in[1,n_i-1]\wedge k\neq j\}}|j \in [1, n_i - 1]\}, \alpha^{r_1 \cdots r_{n_i-1}}$ to u.
3. u chooses a random exponent r_u and computes $\{\alpha^{r_u \prod\{r_k|k\in[1,n_i-1]\wedge k\neq j\}}|j \in [1, n_i - 1]\}$, $\alpha^{r_u r_1 \cdots r_{n_i-1}}$, and sends them to the leader D_i.
4. D_i chooses an exponent r_{new} and computes $\{\alpha^{r_{new} r_u \prod\{r_k|k\in[1,n_i-1]\wedge k\neq j\}}|j \in [1, n_i - 1]\}$, $\alpha^{r_{new} r_1 \cdots r_{n_i-1}}$, and broadcast to all the members in the cluster.

All the members in the cluster get the cluster key $\alpha^{r_{new} r_u r_1 \cdots r_{n_i-1}}$.

4.2 Key Refreshing

Key refreshing in mobile ad hoc networks is of great importance, because most nodes can be easily compromised due to their mobility and physical vulnerability. After the group key (the cluster key) is used for a period of time, the members in the group (the cluster) need refresh the group (the cluster) key in order to limit exposure due to the loss of group (cluster) session keys and limiting the amount of ciphertext available to cryptanalysis for a given group (cluster) session key.

Key refreshing protocol is based on the refreshment of member's key piece. Let D_i be the dominator of the cluster C_i and the random number he holds is r_D. When the cluster key should be refreshed, the key refreshing protocol is done as follows:

1. D_i chooses a new random number r'_D.
2. D_i computes $\{\alpha^{r'_D \prod\{r_k|k\in[1,n_i]\wedge k\neq j\}}|j \in [1,n_i]\}$, and broadcast to all the members in the cluster.

So now the key piece hold by the member in the cluster is $\alpha^{r'_D \prod\{r_k|k\in[1,n_i]\}}$.

When the group key should be refreshed, all the cluster leader process the protocol similar to above such that each of them gets the new group key. Then each cluster leader distributes the group key to all the members in the cluster.

4.3 Member Leaving

Deleting a mobile node M_i from the mobile ad hoc networks is also easy in our hierarchical key agreement protocol. Here, we assume that the leaving of node M_i will not disconnect the network.

If M_i is a member in the cluster C_l whose leader is node D_l, then D_l just apply member deletion [17] to get new cluster key for the cluster C_l. That is, D_l chooses a random number r_{new} and computes a new set of $n_l - 2$ sub-keys: $\{\alpha^{\prod\{r_j|j\in[1,n_l]\wedge j\neq k\}}|k \neq i\}$. Then he broadcast them to all cluster number so that all cluster number get a new cluster key. Next, all the cluster leaders process the key refreshing protocol to get the new group key, and each cluster leader distributes the new group key to all the members in the cluster.

Assume M_i is the leader of the cluster C_l. The protocol is done as follows:

1. M_{i-1} will take place of the leader of the cluster C_l and join to the backbone network.
2. All the member in cluster C_l using member deletion protocol in [17] to get a new cluster key.
3. The last member of the backbone network (D_g, if M_i is not the last member, and D_{g-1}, if M_i is the last number) sends $\{\alpha^{\prod\{R_k|k\in[1,g]\wedge k\neq j\}}|j \neq i\}$ to M_{i-1}, where R_k is the random number chosen by D_k at the key establishment phase.
4. M_{i-1} chooses a random number R' and computes $\{\alpha^{R' \prod\{R_k|k\in[1,g]\wedge k\neq j\}}|j \neq i\}$. Then he broadcast this to all the member in the backbone network so that all the cluster leaders get a new group key.
5. Each cluster leader distributes the group key to all the members in the cluster.

4.4 Analysis

When we execute the member joining, key refreshing or member leaving protocol, the first step is to find the route of the clusters and the backbone network. It can be done by lots of routing protocol such as [10] and the communication cost of the this step is $O(n)$.

After the key establishment, the nodes in the mobile ad hoc network moves around in the network. After a period of time, all the members in a cluster may disperse into the network, and we can view the cluster as the backbone networks. For the GDH protocol is used for the key agreement among each cluster nodes, we first find members in the cluster and then order the cluster nodes and connectors as $M_{i,1}$, $M_{i,2}$, ..., M_{i,n_i}, such that the total communication cost from $M_{i,j}$ to $M_{i,j+1}$ $(1 \leq j < n_i)$, is linear. Notice that there are n_i nodes totally. For simplicity, let $c(M_{i,j}, M_{i,j+1})$ be the communication cost from $M_{i,j}$ to $M_{i,j+1}$ in the network, i.e., the number of hops connecting them. We analyze the total communication cost $\sum c(M_{i,j}, M_{i,j+1})$ for the ordering derived by the above method. For the communication cost from $M_{i,j}$ to $M_{i,j+1}$ is linear, the total communication cost for broadcasting a message in a cluster is at most $\sum c(M_{i,j}, M_{i,j+1})$, which will be $O(n_i)$. Thus, the total communication cost for all cluster leader distributes the group key to the cluster members in the network is $\sum_i$ (the total communication cost for broadcasting a message in a cluster C_i) $= \sum_i O(n_i) = O(n)$. Then, we can obtain the following table:

		joining	refreshing (group)	leaving (dominator)
comm. cost	hierarchical	$2c_1 + c_2$	$O(n) + c_2$	$c_1 + 2c_2 + O(n)$
	original GDH	$c_1 + n$	n	n
lasting time	hierarchical	$2t_1 + t_2$	$2t_2$	$t_1 + 3t_2$
	original GDH	$t_1 + t_2$	t_2	t_2
total exp.	hierarchical	$3b$	$2g$	$2b + 2g$
	original GDH	$2n$	$2n$	$2n$

From this table, we can see that our protocol does not increase much communication cost and lasting time compare with the original GDH protocol, and total exponentiation is much lower than the original GDH protocol.

5 Conclusion

In this paper, we propose a hierarchical key agreement protocol that is communication efficient to mobile ad hoc networks. We also show how to manage the group efficiently in mobile environments. Our protocol has following advantages:

1. The member can send encrypted messages to all using the group key and send encrypted messages to the members in its cluster using the cluster key.
2. The protocol is efficient during new members joining in, refreshing the group key and deleting the leaving members.

3. Since the nodes are mobile, the network topology may change rapidly and unpredictably and the connectivity among the terminals may vary with time. Our key agreement protocol does not maintain the topology of the network, and need not change the group key when the member are moving.
4. Because there are not many nodes in each cluster, it is easy to transmit a password and use the authenticated key agreement protocol.

Acknowledgements. This work is supported by the National Grand Fundamental Research 973 Program of China under Grant No. G1999035802; the Youth Foundation of the National Natural Science of China under Grant No. 60025205; the research collaboration program between Institute for InfoComm Research, Singapore and Institute of Software, Chinese Academy of Sciences. The authors would like to thanks the anonymous referees for their helpful comments.

References

1. N. Asokan and P. Ginzboorg, "Key-Agreement in Ad-hoc Networks", *Computer Communications*, vol. 23, no. 17, pp. 1627–1637, 2000.
2. G. Ateniese, M. Steiner and G. Tsudik, "Authenticated group key agreement and friends", *ACM Conference on Computer and Communications Security — CCS'98*, pp. 17–26, 1998.
3. K. Alzoubi, P.-J. Wan and O. Frieder, "Message-optimal connected-dominating-set construction for routing in mobile ad hoc networks", *ACM International Symposium on Mobile Ad Hoc Networking and Computing — MobiHoc'02*, 2002.
4. M. Burmester, "On the risk of opening distributed keys", *Advances in Cryptology — CRYPTO'94, LNCS 839*, Springer-Verlag, pp. 308–317, 1994.
5. Y. Desmedt and M. Burmester, "Towards practical proven secure authenticated key distribution", *ACM Conference on Computer and Communications Security — CCS'93*, pp. 228–231, 1993.
6. M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system", *Advances in Cryptology — EUROCRYPT'94, LNCS 950*, pp. 275–286, 1994.
7. K. Becker and U. Wille, "Communication complexity of group key distribution", *ACM Conference on Computer and Communications Security —CCS'98*, pp. 1–6, 1998.
8. W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.
9. M. Gerla and J. T. Tsai, "Multicluster, mobile, multimedia radio network", *ACM-Baltzer Journal of Wireless Networks*, vol.1, no.3, pp. 255–265, 1995.
10. X. Hong, K. Xu and M. Gerla, "Scalable routing protocols for mobile ad hoc networks", *IEEE Network*, vol. 16, no. 4, pp. 11–21, 2002.
11. Y. Kim, A. Perrig and G. Tsudik, "Tree-based group key agreement", *Cryptology ePrint Archive, Report 2002/009*, 2002.
12. Y. Kim, A. Perrig and G. Tsudik, "Simple and Fault-Tolerant Key Agreement For Dynamic Collaborative Groups", 7th ACM Conference on Computer and Communications Security, pp. 235–244, 2000.

13. C. R. Lin and M. Gerla, " Adaptive clustering for mobile networks", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.
14. X. Li, Y. Wang and O. Frieder, "Efficient Hybrid Key Agreement Protocol for Wireless Ad Hoc Networks", *IEEE International Conference on Computer Communications and Networks — ICCCN 2002*, 2002.
15. A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
16. G. Pei, M. Gerla, X. Hong and C.-C. Chiang, "A wireless hierarchical routing protocol with group mobility", *IEEE Wireless Communications and Networking Conference — WCNC'99*, 1999.
17. M. Steiner, G. Tsudik and M. Waidner, "Diffie-Hellman key distribution extended to group communication", *ACM Conference on Computer and Communications Security — CCS'96*, pp. 31–37, 1996.
18. M. Steiner, G. Tsudik and M. Waidner, "CLIQUES: A new approach to group key agreement", *International Conference on Distributed Computing Systems — 1998*, pp. 380–387, 1998.
19. P.-J. Wan, K. M. Alzoubi and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks", *Annual Joint Conference of the IEEE Computer and Communications Societies — INFOCOM 2002*, 2002.
20. Y. Wang and X. Li, "Geometric spanners for wireless ad hoc networks", *IEEE International Conference on Distributed Computing Systems — ICDCS 2002*, 2002.
21. K. Xu, X. Hong and M. Gerla, "An ad hoc network with mobile backbones", *IEEE International Conference on Communications — ICC 2002*, 2002.

An Efficient Tree-Based Group Key Agreement Using Bilinear Map

Sangwon Lee[1], Yongdae Kim[2], Kwangjo Kim[1], and Dae-Hyun Ryu[3]

[1] Information and Communications University (ICU),
58-4, Hwaam-Dong, Yuseong-gu, Daejeon, 305-732, Korea,
{swlee,kkj}@icu.ac.kr
[2] University of Minnesota - Twin cities
4-192 EE/CSci Building, 200 Union Street S.E., Minneapolis, MN 55455
kyd@cs.umn.edu
[3] Hansei University,
604-5, Dangjung-dong, Kunpo-si, Kyunggi-do, Seoul, 435-742, Korea,
dhryu@hansei.ac.kr

Abstract. Secure and reliable group communication is an increasingly active research area by growing popularity in group-oriented and collaborative application. One of the important challenges is to design secure and efficient group key management. While centralized management is often appropriate for key distribution in large multicast-style groups, many collaborative group settings require distributed key agreement. The communication and computation cost is one of important factors in the group key management for Dynamic Peer Group. In this paper, we extend TGDH (Tree-based Group Diffie-Hellman) protocol to improve the computational efficiency by utilizing pairing-based cryptography. The resulting protocol reduces computational cost of TGDH protocol without degrading the communication complexity.

Keywords: Group key agreement, TGDH, Bilinear Diffie-Hellman, Bilinear map, Pairings

1 Introduction

Secure and reliable communications have become critical in modern computing. The centralized services like e-mail and file sharing can be changed into distributed or collaborated system through multiple systems and networks. Basic cryptographic requirements such as data confidentiality, data integrity, authentication and access control are required to build secure collaborative system in the broadcast channel. When all group members have the shared secret key, these security services can be easily implemented.

Dynamic Peer Group (DPG) belongs to a kind of *ad hoc* group which its membership can be frequently changed and the communicating party in a group can be dynamically configured.

Recently, Joux[5] presented a three-party key agreement protocol which requires each entity to make on a single round using pairings on algebraic curves.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 357–371, 2003.

This should be contrasted with the obvious extension of the conventional Diffie-Hellman key distribution protocol to three parties requiring two interactions per peer entity. We extend this three-party key agreement protocol to group key agreement protocol using ternary tree and also use two-party key agreement protocol for some subtree node.

Y. Kim *et al.*[9] proposed a secure, simple and efficient key management method, called TGDH(Tree-based Group Diffie-Hellman) protocol, which uses key tree with Diffie-Hellman key exchange to efficiently compute and update group keys. Since the computation cost of tree-based key management is proportional to the height of configured key tree. Using ternary key tree, we can reduce the computation cost $O(log_2 n)$ of TGDH to $O(log_3 n)$.

This paper is organized as follows: Section 2 briefly introduces previous work in group key management, group membership events and bilinear map. Section 3 explains the protocol. Performance analysis is described in Section 4. We suggest concluding remarks in Section 5 following with the security analysis of our protocol in Appendix.

2 Previous Work

2.1 Group Membership Operations

A comprehensive group key agreement must handle adjustments to group secrets subsequent to all membership operations in the underlying group communication system.

We distinguish among single and multiple member operations. Single member changes include member addition or deletion. This occurs when a member wants to join(or leave) a group. Multiple member changes also include addition and deletion: ***Member Join*** and ***Leave***. We refer to the multiple addition operation as ***Group Merge***, in which case two or more groups merge to form a single group. We refer to the multiple leave operation as ***Group Partition***, whereby a group is split into smaller groups. ***Group Merge*** and ***Partition*** event are common owing to network misconfiguration and router failures. Hence, dealing with ***Group Partition*** and ***Merge*** is a crucial component of group key agreement.

In addition to the single and multiple membership operations, periodic refreshes of group secrets are advisable so as to limit the amount of ciphertext generated with the same key and to recover from potential compromise of member's contribution or prior session keys. ***Key Refresh*** is one of the most important security requirements of a group key agreement.

The special member, referred to as *sponsor*, is responsible for broadcasting all link values of the current tree to the members. Note that the *sponsor* is not a privileged member. His task is only to broadcast the current tree information to the group members. Any current member could perform this task. We assume that every member can unambiguously determine both the *sponsors* and the insertion location in the key tree. ***Key Refresh*** operation can be considered to be a special case of ***Member Leave*** without any members actually leaving the group.

Group key agreement of dynamic group must provide four security properties: Group key secrecy is basically supported property in group communication. Forward secrecy means that any leaving member from a group can not generate new group key. Backward secrecy means that any joining member into a group can not discover previously-used group key. The combination of backward secrecy and forward secrecy forms key independence.

2.2 Bilinear Pairings and BDH Assumption

Let G_1 be an additive group generated by P, whose order is a prime q, and G_2 be a multiplicative group of the same order q. We assume that the discrete logarithm problem(DLP) in both G_1 and G_2 is hard. Let $e : G_1 \times G_1 \rightarrow G_2$ be a paring which satisfies the following conditions:

1. Bilinear: $e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)$ and $e(P, Q_1 + Q_2) = e(P, Q_1)$ $e(P, Q_2)$
2. Non-degenerate : The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2. Observe that since G_1, G_2 are groups of prime order this implies that if P is a generator of G_1 then $\widehat{e}(P, P)$ is a generator of G_2.
3. Computability : There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$

The Weil or Tate pairings associated with supersingular elliptic curves or Abelian varieties can be modified to create such bilinear maps.

BDH Problem: The Bilinear Diffie-Hellman(BDH) Problem for a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ is defined as follows: given $P, aP, bP, cP \in G_1$, compute $e(P, P)^{abc}$, where a, b, c are randomly chosen from Z_q^*. An algorithm $\mathcal{A}$ is said to solve the BDH problem with an advantage of ϵ if

$$Pr[\mathcal{A}(P, aP, bP, cP) = e(P, P)^{abc}] \geq \epsilon$$

BDH Assumption: We assume that the BDH problem is hard, which means there is no polynomial algorithm to solve BDH problem with non-negligible probability.

3 Our Protocol

Table 1 shows the notations used in this paper. We can classify three nodes of a key tree as follows:

- Member node : represent each group member as leaf node.
- Key node : correspond with one key. This key is shared by all members of the subtree rooted at this key node.
- Root node : represent the shared group key.

Table 1. Notations

N	Member of protocol parties(group members)
C	Set of current group members
L	Set of leaving members
M_i	i-th group member; $i \in \{1,2, ..., N\}$
h	The height of the key tree
$< l, v >$	v-th node at the l-th level in a tree
T_i	M_i's view of the key tree
$\widehat{T}_i$	M_i's modified tree after membership operation
$T_{<i,j>}$	A subtree rooted at node $< i, j >$
BK_i^*	set of M_i's blinded keys
P	Public information, a point on an elliptic curve
H_1	Hash function, $H_1 : G_2 \rightarrow Z_q^*$
H_2	Hash function, $H_2 : G_1 \rightarrow Z_q^*$

Fig. 1 shows an example of a key tree. The root is located at the 0-th level and the lowest leaves are at the h-th level. Since we use ternary tree, every node can be a leaf or a parent of two nodes or a parent of three nodes. The node are denoted $< l, v >$, where $0 \leq v \leq 3^l - 1$ since each level l hosts at most 3^l nodes. Each node $< l, v >$ is associated with the *key* $K_{<l,v>}$ and the blinded key (*bkey*) $BK_{<l,v>} = K_{<l,v>}P$. The multiplication kP is obtained by repeating k times addition over an elliptic curve. We assume that a leaf node $< l, v >$ is associated with M_i, then the node $< l, v >$ has M_i's session random key $K_{<l,v>}$. We further assume that the member M_i at node $< l, v >$ knows every key along the path from $< l, v >$ to $< 0, 0 >$, referred to as the *key-path*. In Fig. 1, if a member M_3 owns the tree T_3, then M_3 knows every *key* $\{K_{<2,2>}, K_{<1,0>}, K_{<0,0>}\}$ and every *bkey* $BK_3^* = \{BK_{<2,2>}, BK_{<1,0>}, BK_{<0,0>}\}$ on T_3.

The case of subtree having three child node at $< l, v >$, computing a key requires the knowledge of the *key* in one of the three child node and the *bkey* of the other child node. We can get a *key* $K_{<l,v>}$ by computing pairings. In another case, we need to know the *key* of one of the two child node and the *bkey* of the other child node. We can get a *key* $K_{<l,v>}$ by computing a point multiplication on elliptic curve. $K_{<0,0>}$ at the root node is the group secret shared by all members.

For example, in Fig. 1, M_3 can compute $K_{<1,0>}$, $K_{<0,0>}$ using $BK_{<2,0>}$, $BK_{<2,1>}$, $BK_{<1,1>}$ and $K_{<2,2>}$. The final group key $K_{<0,0>}$ is :

$$K_{<0,0>} = H_1(\widehat{e}(H_1(\widehat{e}(P,P)^{r_4 r_5 r_6})P, r_7 P)^{H_1(\widehat{e}(r_1 P, r_2 P)^{r_3})})$$

If there are 8 members in group, then the final group key $K_{<0,0>}$ is :

$$K_{<0,0>} = H_1(\widehat{e}(H_1(\widehat{e}(P,P)^{r_4 r_5 r_6})P, H_2(r_7 r_8 P)P)^{H_1(\widehat{e}(r_1 P, r_2 P)^{r_3})})$$

where $r_7 r_8 P$ is the shared key between M_7 and M_8 using ECDH (Elliptic Curve Diffie-Hellman)problem.

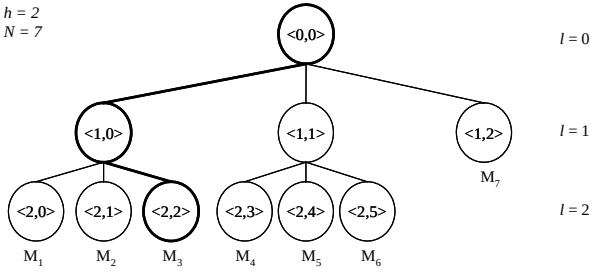

Fig. 1. An example of a key tree

Now we describe the group operation: ***Join***, ***Leave***, ***Partition*** and ***Merge***. We modify this operation in TGDH by utilizing the ternary tree and bilinear map.

3.1 Join Protocol

We assume the group has n members: $\{M_1, M_2, ..., M_n\}$. The new member M_{n+1} initiates the protocol by broadcasting a join request message that contains its own *bkey* $BK_{<0,0>}$ $(= r_{n+1}P)$.

Each current member receives this message and first determines the insertion point in the tree. The insertion point is the shallowest rightmost node, where the join does not increase the height of the key tree. Otherwise, if the key tree is fully balanced, the new member joins to the root node. The *sponsor* is the rightmost leaf in the subtree rooted at the insertion point. If the intermediate node in the rightmost has two member nodes, the *sponsor* inserts the new member node under this intermediate node. The tree becomes fully balanced. Otherwise, each member creates a new intermediate node and a new member node, and promotes the new intermediate node to be the parent of both the insertion node and the new member node. After updating the tree, all members, except the *sponsor*, are blocked. The *sponsor* proceeds to update his share and computes the new group key; the *sponsor* can do this operation since it knows all necessary *bkeys*. Next, the *sponsor* broadcasts the new tree which contains all *bkeys*. All other members update their trees accordingly and compute the new group key.

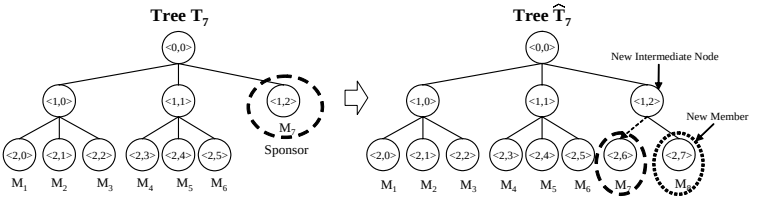

Fig. 2. Tree-updating in join operation

Table 2. Join Protocol

Step 1 : The new member broadcasts request for join

$$M_{n+1} \xrightarrow{BK_{<0,0>}=r_{n+1}P} C$$

Step 2 : Every member
- if key tree contains the subtree that has two child node, add the new member node for updating key tree. otherwise, add the new member node and new intermediate node,
- remove all *keys* and *bkeys* from the leaf node related to the *sponsor* to the root node.

The *sponsor* M_s additionally
- generates new share and computes all $[key, bkey]$ pairs on the *key-path*,
- broadcasts updated tree $\widehat{T}_s$ including only *bkeys*.

$$M_s \xrightarrow{\widehat{T}_s(BK_s^*)} C \cup \{M_{n+1}\}$$

Step 3 : Every member computes the group key using $\widehat{T}_s$.

It might appear wasteful to broadcast the entire tree to all members, since they already know most of the *bkeys*. However, since the *sponsor* needs to send a broadcast the entire tree to the group anyhow, it might as well include more information which is useful to the new member, thus saving one unicast message to the new member (which would have to contain the entire tree).

Fig. 2 illustrates an example of member M_8 joining a group where the *sponsor* (M_7) performs the following actions:

1. Rename node $< 1, 2 >$ to $< 2, 6 >$.
2. Generate a new intermediate node $< 1, 2 >$ and a new member node $< 2, 7 >$.
3. Update $< 1, 2 >$ as the parent node of $< 2, 6 >$ and $< 2, 7 >$.
4. Generate new share and compute all $[key, bkey]$ pairs.
5. Broadcast updated tree $\widehat{T}_7$.

Since all members know $BK_{<2,7>}$, $BK_{<1,0>}$ and $BK_{<1,1>}$, M_7 can compute the new group key $K_{<0,0>}$. Every other member also performs steps 1 and 2, but cannot compute the group key in the first round. Upon receiving the broadcasted *bkeys*, every member can compute the new group key.

If another member M_9 wants to join the group, the new *sponsor*(M_8) performs the following actions:

1. Generate a new member node $< 2, 8 >$ under the intermediate node $< 1, 2 >$.
2. Generate new share and compute all $[key, bkey]$ pairs.
3. Broadcast updated tree $\widehat{T}_8$.

Every member also performs step 1, and then can compute the new group key with the broadcasted messages.

3.2 Leave Protocol

Such as ***Join*** protocol, we start with n members and assume that member M_d leaves the group. The *sponsor* in this case is the rightmost leaf node of the subtree rooted at leaving member's sibling node. First, if the number of leaving member's sibling node is two, each member updates its key tree by deleting the leaf node corresponding to M_d. Then the former sibling of M_d is updated to replace M_d's parent node. Otherwise each member only deleting the leaf node corresponding to M_d. The *sponsor* generates a new key share, computes all $[key, bkey]$ pairs on the *key-path* up to the root, and broadcasts the new set of *bkey*. This allows all members to compute the new group key. In Fig. 3, if member M_7 leaves the group, every remaining member deletes $< 1, 2 >$ and $< 2, 6 >$. After updating the tree, the *sponsor* (M_{10}) picks a new share $K_{<2,8>}$, recomputes $K_{<1,2>}$, $K_{<0,0>}$, $BK_{<2,8>}$ and $BK_{<1,2>}$, and broadcasts the updated tree $\widehat{T}_{10}$ with BK^*_{10}. Upon receiving the broadcast message, all members compute the group key. Note that M_7 cannot compute the group key, though he knows all the *bkeys*, because his share is no longer a part of the group key.

Table 3. Leave Protocol

Step 1 : Every member
- update key tree by removing the leaving member node,
- remove relevant parent node, if this node have only one member node,
- remove all *keys* and *bkeys* from the leaf node related to the *sponsor* to the root node.

The *sponsor* M_s additionally
- generates new share and computes all $[key, bkey]$ pairs on the *key-path*,
- broadcasts updated tree $\widehat{T}_s$ including only *bkeys*.

$$M_s \xrightarrow{\widehat{T}_s(BK^*_s)} C - L$$

Step 2 : Every member computes the group key using $\widehat{T}_s$.

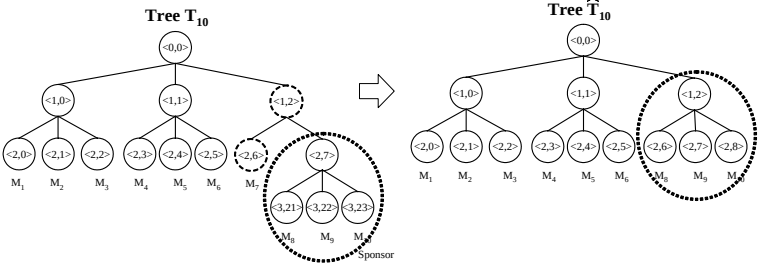

Fig. 3. Tree-updating in leave operation

Table 4. Partition Protocol

Step 1 : Every member
- update key tree by removing all the leaving member node,
- remove their relevant parent node, if this node have only one member node,
- remove all *keys* and *bkeys* from the leaf node related to the *sponsor* to the root node.

Each *sponsor* M_{s_t}
- if M_{s_t} is the shallowest rightmost *sponsor*, generate new share,
- compute all $[key, bkey]$ pairs on the *key-path* until it can proceed,
- broadcast updated tree $\widehat{T}_{s_t}$ including only *bkeys*.

$$M_{s_t} \xrightarrow{\widehat{T}_{s_t}(BK^*_{s_t})} C - L$$

Step 2 to h (Until a *sponsor* M_{s_j} could compute the group key)
: For each *sponsor* M_{s_t}
- compute all $[key, bkey]$ pairs on the *key-path* until it can proceed,
- broadcast updated tree $\widehat{T}_{s_t}$ including only *bkeys*.

$$M_{s_t} \xrightarrow{\widehat{T}_{s_t}(BK^*_{s_t})} C - L$$

Step $h + 1$: Every member computes the group key using $\widehat{T}_s$.

In Fig. 3, if member M_{10} leaves the group, every remaining members delete only $< 3, 23 >$. After updating the tree, the *sponsor* (M_9) generates new share $K_{<3,22>}$, recomputes $K_{<2,7>}$, $K_{<1,2>}$, $K_{<0,0>}$, $BK_{<2,7>}$ and $BK_{<1,2>}$, and broadcasts the updated tree $\widehat{T}_9$ with BK^*_9. Upon receiving the broadcast message, all members can compute the group key.

3.3 Partition Protocol

We assume that a network failure causes a partition of the n-member group. From the viewpoint of each remaining member, this event appears as a simultaneous leaving of multiple members. The ***Partition*** protocol is involves multiple rounds; it runs until all members compute the new group key. In the first round, each remaining member updates its tree by deleting all partitioned members as well as their respective parent nodes and "compacting" the tree. The procedure is summarized in Table 4.

Fig. 4 shows an example. In the first round, all remaining members delete all nodes of leaving members and compute *keys* and *bkeys*. Any member can not compute the group key since they lack the *bkey* information. However, M_5 generates new share and computes and broadcasts $BK_{<1,0>}$ in the first round, and M_{13} can thus compute the group key. After M_{13} generates new share and

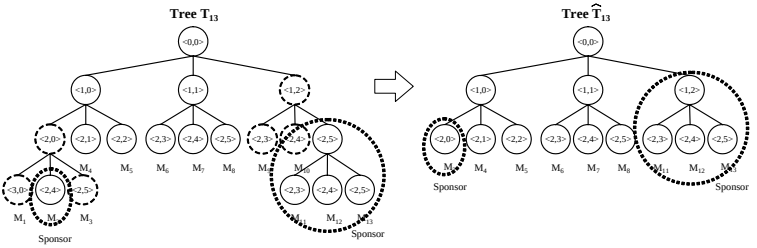

Fig. 4. Tree-updating in partition operation

broadcasts $BK_{<1,2>}$, M_5 can compute the group key. Finally every member knows all *bkeys* and can compute the group key.

Note that if some member M_i can compute the new group key in round h', then all other member can compute the group key, in round $h' + 1$, since M_i's broadcast message contains every *bkey* in the key tree, each member can detect the completion of the partition protocol independently.

3.4 Merge Protocol

After the network failure recovers, subgroup may need to be merged back into a single group. We now describe the merge protocol for k merging groups.

In the first round of the merge protocol, each *sponsor*(the rightmost member of each group) broadcasts its tree with all *bkeys* to all other groups after updating the secret share of the *sponsor* and relevant $[key, bkey]$ pairs up to the root node. Upon receiving these message, all members can uniquely and independently determine how to merge those k trees by tree management policy.

Next, each *sponsor* computes $[key, bkey]$ pairs on the *key-path* until either this computation reaches the root or the *sponsor* can not compute a new intermediate key. The *sponsor* broadcast his view of the tree to the group. All members then update their tree views with the new information. If the broadcasting *sponsor* computed the root key, upon receiving the broadcast, all other members can compute the root key as well.

Fig. 5 shows an example of merging two groups, where the *sponsors* M_5 and M_{14} broadcast their trees (T_5 and T_{14}) containing all the *bkeys*, along with BK_5^* and BK_{14}^*. Upon receiving these broadcast messages, every member checks whether it belongs to the *sponsor* in the second round. Every member in both groups merges two trees, and then the *sponsor*(M_5) in this example updates the key tree and computes and broadcasts *bkeys*.

4 Performance

This section analyzes the communication and computation costs for ***Join***, ***Leave***, ***Merge*** and ***Partition*** protocols. We count the number of rounds, the total number of messages, the serial number of exponentiations, pairings and

Table 5. Merge Protocol

Step 1 : All *sponsors* M_{s_i} in each T_{s_i}
- generate new share and compute all $[key, bkey]$ pairs on the *key-path* of T_{s_i},
- broadcast updated tree $\widehat{T}_{s_i}$ including only *bkeys*.

$$M_{s_i} \xrightarrow{\widehat{T}_{s_i}(BK^*_{s_i})} \bigcup_{i=1}^{k} C_i$$

Step 2 : Every member
- update key tree by adding new trees and new intermediate nodes,
- remove all *keys* and *bkeys* from leaf node related to the *sponsor* to the root node.

Each *Sponsor* M_{s_t} additionally
- compute all $[key, bkey]$ pairs on the *key-path* until it can proceed,
- and broadcast updated tree $\widehat{T}_{s_t}$ including only *bkeys*.

$$M_{s_t} \xrightarrow{\widehat{T}_{s_t}(BK^*_{s_t})} \bigcup_{i=1}^{k} C_i$$

Step 3 to h (Until a *sponsor* M_{s_j} could compute the group key)
: For each *sponsor* M_{s_t}
- computes all $[key, bkey]$ pairs on the *key-path* until it can proceed,
- and broadcasts updated tree $\widehat{T}_{s_t}$ including only *bkeys*.

$$M_{s_t} \xrightarrow{\widehat{T}_{s_t}(BK^*_{s_t})} \bigcup_{i=1}^{k} C_i$$

Step $h+1$: Every member computes the group key using $\widehat{T}_s$

point multiplications. The serial cost assumes parallelization within each protocol round and presents the greatest cost incurred by any participant in a given round(or protocol). The total cost is simply the sum of all participants' costs in a given round(or protocol).

Table 6 summarizes the communication and computation costs of TGDH and our protocol. The number of current group members, merging groups and leaving members are denoted by n, k and p, respectively. The overhead of protocol depends on the tree height, the balance of the key tree, the location of the joining tree and the leaving nodes. In our analysis, we assume the worst case configuration and list the worst-case cost for TGDH and our protocol.

Since we modified TGDH protocol, the number of communication is equals to TGDH except the number of rounds in merge and key length. But our proposed protocol can reduce the number of computation in each event operation because of low height of key tree. The number of pairings and point multiplications for

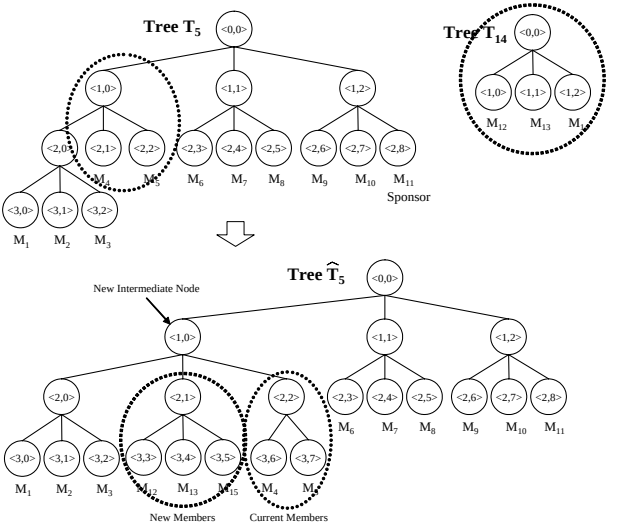

Fig. 5. Tree-updating in merge operation

Table 6. Communication and Computation Costs

		Communication		Computation		
		Rounds	Messages	Exponentiations	Pairings	Multiplications
TGDH	Join	2	3	$\frac{3}{2}\lceil log_2 n \rceil$	0	0
	Leave	1	1	$\frac{3}{2}\lceil log_2 n \rceil$	0	0
	Merge	$log_2 k + 1$	$2k$	$\frac{3}{2}\lceil log_2 n \rceil$	0	0
	Partition	$min(log_2 p, h)$	$2\lceil log_2 n \rceil$	$3\lceil log_2 n \rceil$	0	0
Our Protocol	Join	2	3	0	$\lceil log_3 n \rceil - 1$	$\lceil log_3 n \rceil + 1$
	Leave	1	1	0	$\lceil log_3 n \rceil - 1$	$\lceil log_3 n \rceil + 1$
	Merge	$log_3 k + 1$	$2k$	0	$\lceil log_3 n \rceil - 1$	$\lceil log_3 n \rceil + 1$
	Partition	$min(log_3 p, h)$	$2\lceil log_3 n \rceil$	0	$2\lceil log_3 n \rceil$	$2\lceil log_3 n \rceil$

our protocol depends on whether there exists the subtree with two member nodes or not. We thus compute the cost of average case.

In all events we can reduce the computation cost $O(log_2 n)$ to $O(log_3 n)$. We can get the advantage of the number of computation about 4 times in ***Join***, ***Leave*** and ***Merge*** and 2 times in ***Partition***. The pairings computation is a critical operation in pairings based cryptosystem. The research of pairings implementation continuously have been studied. Barreto *et al.*[3] proposed an efficient algorithm for pairing-based cryptosystems. In this research we can get the result that computing pairings is about 3 times slower than the modular exponentiation. Therefore our protocol requires less the number of communication and computation than TGDH. However, since involving the pairings computation, our protocol admits of improvement in computational efficiency.

The security analysis of our protocol is in Appendix for details. We describe and prove the Decisional Ternary tree Group Bilinear Diffie-Hellman (DTGBDH) problem.

5 Concluding Remarks

This paper present TGDH group event operation using bilinear map. The modified TGDH using bilinear map support dynamic membership group events with forward and backward secrecy. Our protocol involves pairings operation whose computation is computationally slower than modular exponentiation. However, fast implementation of pairings has been studied actively recently. Since we use ternary key tree, our protocol can use any two-party and three-party key agreement protocol. In this paper, because we use the two-party key agreement protocol using ECDH and the three-party key agreement protocol using bilinear map, the security of our protocol relies on this two protocol. Finally our protocol can reduce the number of computation in group events while preserving the communication and the security property.

References

1. S. Al-Riyami and K. Paterson, "Authenticated three party key agreement protocols from pairings," Cryptology ePrint Archive, Report 2002/035, available at http://eprint.iacr.org/2002/035/.
2. D. Boneh and M. Franklin. "Identity-based encryption from the Weil pairing," Advances in Cryptology-Crypto 2001, LNCS 2139, pp.213–229, Springer-Verlag, 2001. http://www.crypto.stanford.edu/ dabo/abstracts/ibe.html
3. P.S.L.M. Barreto, H.Y. Kim, B.Lynn, and M.Scott, "Efficient Algorithms for pairing-based cryptosystems," To appear in Cryptology-Crypto'2002, available at http://eprint.iacr.org/2002/008/.
4. Wallner, Debby M., Eric J. Harder, and Ryan C. Agee, "Key management for multicast: Issues and architectures," RFC 2627, June 1999.
5. A. Joux, "A one round protocol for tripartite Diffie-Hellman," In W. Bosma, editor, Proceedings of Algorothmic Number Theory Symposium – ANTS IV, volume 1838 of LNCS, pages 385–394. Springer-verlag, 2000
6. A. Joux, "The Weil and Tate Pairings as building blocks for public key cryptosystems," in Algorithm Number Theory, 5th International Symposium ANTS-V, LNCS 2369, Springer-Verlag, 2002, pp. 20–32.
7. N. Koblitz, "Elliptic curve cryptosystems," Mathematics of Computation, vol. 48, pp. 203–209, 1987
8. Y. Kim. A. Perrig and G. Tsudik, "Communication-Efficient Group Key Agreement," IFIP SEC 2001, Jun. 2001.
9. Y. Kim, A. Perrig, G. Tsudik, "Tree-based Group Diffie-Hellman Protocol," ACM-CCS 2000.
10. A. Perrig, D. Song, and J. D. Tyger, "ELK, a New Protocol for Efficient Large Group Key Distribution," In 2001 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 2001.

11. N.P. Smart, "An identity based authenticated key agreement protocol based on the weil pairing," Election. Lett., Vol.38, No.13, pp.630–632, 2002
12. F. Zhang, S. Liu and K. Kim, "ID-Based One Round Authenticated Tripartite Key Agreement Protocol with Pairings," Available at http://eprint.iacr.org, 2002.

Appendix Security Analysis

Here we describe Decisional Ternary tree Group Bilinear Diffie-Hellman (DTGBDH) problem and apply security proof of TGDH in [9] to the ternary key tree.

For $(q, G_1, G_2, \widehat{e}) \leftarrow g(1^k)$, $n \in N$ and $X = (R_1, R_2, ..., R_n)$ for $R_i \in Z_q^*$ and a key tree T with n leaf nodes which correspond to R_i, we define the following random variables:

- K_j^i : i-th level of j-th key (secret value), each leaf node is associated with a member's session random, *i.e.*, $K_j^0 = R_k$ for some $k \in [1, n]$.
- BK_j^i : i-th level of j-th blinded key (public value), *i.e.*, $K_j^i P$.
- K_j^i is recursively defined as follows:

$$\begin{aligned} K_j^i &= \widehat{e}(P,P)^{K_{3j-2}^{i-1} K_{3j-1}^{i-1} K_{3j}^{i-1}} = \widehat{e}(K_{3j-2}^{i-1} P, K_{3j}^{i-1} P)^{K_{3j-1}^{i-1}} \\ &= \widehat{e}(K_{3j-2}^{i-1} P, K_{3j-1}^{i-1} P)^{K_{3j}^{i-1}} = \widehat{e}(K_{3j-1}^{i-1} P, K_{3j}^{i-1} P)^{K_{3j-2}^{i-1}} \end{aligned}$$

Also we can define public and secret values as below:

- $view(h, X, T) := \{K_j^i P$ where j and i are defined according to $T\}$
- $K(h, X, T) := \widehat{e}(P,P)^{K_1^{h-1} K_2^{h-1} K_3^{h-1}}$

Note that $view(h, X, T)$ is exactly the view of the adversary in our proposed protocol, where the final secret key is $K(h, X, T)$. Let the following two random variables be defined by generating $(q, G_1, G_2, \widehat{e}) \leftarrow g(1^k)$, choosing X randomly from Z_q^* and choosing key tree T randomly from all ternary trees having n leaf nodes:

- $A_h := (view(h, X, T), y)$
- $H_h := (view(h, X, T), K(h, X, T))$

Definition 1. *Let $(q, G_1, G_2, \widehat{e}) \leftarrow g(1^k)$, $n \in N$ and $X = (R_1, R_2, ..., R_n)$ for $R_i \in Z_q^*$ and a key tree T with n leaf nodes which correspond to R_i. A_h and H_h defined as above. **DTGBDH algorithm** $\mathcal{A}_\mathcal{T}$ is a probabilistic polynomial time algorithm satisfying, for some fixed $k > 0$ and sufficiently large m:*

$$|Prob[\mathcal{A}_\mathcal{T}(A_h) = \text{"}True\text{"}] - Prob[\mathcal{A}_\mathcal{T}(H_h) = \text{"}True\text{"}]| > \frac{1}{m^k}$$

*Accordingly, **DTGBDH problem** is to find an Ternary Tree DBDH algorithm.*

Theorem 1. *If the three-party DBDH on G_1, G_2 is hard, then there is no probabilistic polynomial time algorithm which can distinguish A_h from H_h.*

Proof. We first note that A_h and H_h can be rewritten as:
If $X_L = (R_1, R_2, ..., R_l)$, $X_C = (R_{l+1}, R_{l+2}, ..., R_m)$ and $X_R = (R_{m+1}, R_{m+2}, ..., R_n)$ where R_1 through R_l are associated with leaf node in the left tree T_L, $R_l + 1$ through R_m are in the center tree T_C and $R_m + 1$ through R_n are in the right tree T_R:

$$\begin{aligned}
A_h &:= (view(h, X, T), y) \\
&= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad BK_1^{h-1}, BK_2^{h-1}, BK_3^{h-1}, y) \\
&= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad K_1^{h-1}P, K_2^{h-1}P, K_3^{h-1}P, y) \\
H_h &:= (view(h, X, T), K(h, X, T)) \\
&= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad BK_1^{h-1}, BK_2^{h-1}, BK_3^{h-1}, \widehat{e}(P,P)^{K_1^{h-1}K_2^{h-1}K_3^{h-1}}) \\
&= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad K_1^{h-1}P, K_2^{h-1}P, K_3^{h-1}P, \widehat{e}(P,P)^{K_1^{h-1}K_2^{h-1}K_3^{h-1}})
\end{aligned}$$

We prove this **theorem** by induction and contradiction. The 3-party DBDH problem in G_1 and G_2 is equivalent to distinguish A_1 from H_1. We assume that A_{h-1} and H_{h-1} are indistinguishable in polynomial time as the induction hypothesis. We further assume that there exist a polynomial algorithm that can distinguish A_h from H_h for a random ternary tree. We will show that this algorithm can be used to distinguish A_{h-1} from H_{h-1} or can be used to solve the 3-party DBDH problem.

We consider the following equations:

$$\begin{aligned}
A_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad K_1^{h-1}P, K_2^{h-1}P, K_3^{h-1}P, y) \\
B_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad rP, K_2^{h-1}P, K_3^{h-1}P, y) \\
C_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad rP, r'P, K_3^{h-1}P, y) \\
D_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad rP, r'P, r''P, y) \\
E_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad rP, r'P, r''P, \widehat{e}(P,P)^{rr'r''}) \\
F_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad rP, r'P, K_3^{h-1}P, \widehat{e}(P,P)^{rr'K_3^{h-1}}) \\
G_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad rP, K_2^{h-1}P, K_3^{h-1}P, \widehat{e}(P,P)^{rK_2^{h-1}K_3^{h-1}}) \\
H_h &= (view(h-1, X_L, T_L), view(h-1, X_C, T_C), view(h-1, X_R, T_R), \\
&\quad K_1^{h-1}P, K_2^{h-1}P, K_3^{h-1}P, \widehat{e}(P,P)^{K_1^{h-1}K_2^{h-1}K_3^{h-1}})
\end{aligned}$$

Since we can distinguish A_h and E_h in polynomial time, we can distinguish at least one of (A_h, B_h),(B_h, C_h),(C_h, D_h),(D_h, E_h),(E_h, F_h),(F_h, G_h) or (G_h, H_h).

A_h **and** B_h**:** Suppose we can distinguish A_h and B_h in polynomial time. We will show that this distinguisher $\mathcal{A}_{AB_h}$ can be used to solve DTGBDH problem with height $h-1$. Suppose we want to decide whether $P'_{h-1} = (view(h-1, X_1, T_1), r_1)$ is an instance of DTGBDH problem or r_1 is a random number. To solve this, we generate trees T_2 and T_3 of height $h-1$ with distribution X_2 and X_3, respectively. Note that we know all secret and public information of T_2 and T_3. Using P'_{h-1} and (T_2, X_2), (T_3, X_3) pairs, we generate the distribution:

$$P'_h = (view(h-1, X_1, T_1), view(h-1, X_2, T_2), view(h-1, X_3, T_3), \\ r_1P, K(h-1, X_2, T_2)P, K(h-1, X_3, T_3)P, y)$$

Now we put P'_h as input of $\mathcal{A}_{AB_h}$. If P'_h is an instance of $A_h(B_h)$, then P'_{h-1} is an instance $H_{h-1}(A_{h-1})$.

B_h **and** C_h**:** We can generate P'_h by the similar method in (A_h,B_h) and then put P'_h as input of $\mathcal{A}_{BC_h}$ which can distinguish B_h and C_h. If P'_h is an instance of $B_h(C_h)$, then P'_{h-1} is an instance $H_{h-1}(A_{h-1})$.

C_h **and** D_h**:** We can generate P'_h by the similar method in (A_h,B_h) and then put P'_h as input of $\mathcal{A}_{CD_h}$ which can distinguish C_h and D_h. If P'_h is an instance of $C_h(D_h)$, then P'_{h-1} is an instance $H_{h-1}(A_{h-1})$.

D_h **and** E_h**:** Suppose we can distinguish D_h and E_h in polynomial time. Then, this distinguisher $\mathcal{A}_{DE_h}$ can be used to solve 3-party BDH problem in groups G_1 and G_2. Note that rP, r_1P and r_2P are independent random variable from $view(h-1, X_L, T_L)$, $view(h-1, X_C, T_C)$ and $view(h-1, X_R, T_R)$. Suppose we want to decide whether $(aP, bP, cP, e(P,P)^{abc})$ is a BDH quadruple or not. To solve this, we generate three tree T_1, T_2 and T_3 of height $h-1$ with distribution X_1, X_2 and X_3 respectively. Now we generate new distribution:

$$P'_h = (view(h-1, X_1, T_1), view(h-1, X_2, T_2), view(h-1, X_3, T_3), \\ aP, bP, cP, \widehat{e}(P,P)^{abc})$$

Now we put P'_h as input of $\mathcal{A}_{DE_h}$. If P'_h is an instance of $D_h(E_h)$, then $(aP, bP, cP, \widehat{e}(P,P)^{abc})$ is an invalid(valid) BDH quadruple.

E_h **and** F_h**:** We can generate P'_h by the similar method in (A_h,B_h) and then put P'_h as input of $\mathcal{A}_{EF_h}$ which can distinguish E_h and F_h. If P'_h is an instance of $E_h(F_h)$, then P'_{h-1} is an instance $A_{h-1}(H_{h-1})$.

F_h **and** G_h**:** We can generate P'_h by the similar method in (A_h,B_h) and then put P'_h as input of $\mathcal{A}_{FG_h}$ which can distinguish F_h and G_h. If P'_h is an instance of $F_h(G_h)$, then P'_{h-1} is an instance $A_{h-1}(H_{h-1})$.

G_h **and** H_h**:** We can generate P'_h by the similar method in (A_h,B_h) and then put P'_h as input of $\mathcal{A}_{GH_h}$ which can distinguish G_h and H_h. If P'_h is an instance of $G_h(H_h)$, then P'_{h-1} is an instance $A_{h-1}(H_{h-1})$. □

A Key Recovery Mechanism for Reliable Group Key Management

Taenam Cho and Sang-Ho Lee

Dept. of Computer Science and Engineering, Ewha Womans University,
11-1 Daehyun-Dong, Seodaemun-Gu, Seoul 120-750, Korea
{tncho, shlee}@ewha.ac.kr

Abstract. Secret group communication can be achieved by encryption messages with a group key. Dynamic groups face the problem of changing the group key whenever members join or leave. One of the solutions to this problem is to send the updated group key to members via rekey messages in a secure manner. The recovery of lost keys consequently becomes important because a member cannot decrypt the group data if he loses these messages. Saving messages and resending them by KDC (Key Distribution Center) not only requires large saving space, but also causes the transmission and decryption of unnecessary keys. Furthermore, the keys in the unsaved messages cannot be recovered. This paper proposes an efficient method for recovering group keys. The group key generation method presented in this paper is simple, enabling us to recover group keys without storing and eliminating the transmission and decryption of useless auxiliary keys.

1 Introduction

One of the most important security issues in group communication is confidentiality. Confidentiality is necessary in limiting user access to data from distributed simulations or secret corporate conferences, as well as in maintaining billing information of commercial images and online Internet broadcast sources. To achieve confidentiality, only allowed users should gain access to data through encrypted communication using a shared group key[16]. Therefore, it requires safe sharing of the group key only among valid members and securing data against invalid members. Secure sharing of keys involves a KDC that distributes updated keys via rekey messages to members as other members join or leave. In such a system, the loss of rekey messages would disable the reception of the updated group keys, causing an inability to decrypt group data and possibly the failure of decryption of any follow-up rekey messages transmitted. Such reliability issues still remain to be solved[9][10]. Several methods to increase reception rates have been introduced[18][14][19], but none of them address the problem in the recovery of lost keys. Another research[11] proposed a scheme in which a member may calculate the lost keys using verification information. However, this proposal ignores the possibility of the member's loss of the verification information also, in which case he still cannot recover the lost keys. Therefore, more

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 372–386, 2003.

research is required, especially on the problems caused by members' log-in/out and communication delays. This paper proposes a method based on a model in which a KDC generates and distributes keys to members, and also allows the recovery of lost keys when key updates are made after keys have been lost by communication delays or member's log-out.

There are 7 sections in this paper. Section 2 explains the requirements that a group key management must fulfill. Related studies are described in Section 3, and Section 4 proposes a key recovery mechanism. Security, reliability and efficiency analysis are found in Section 5. In section 6, the parameters for optimization are derived. Finally, section 7 presents the conclusion and suggestions for future research.

2 Requirements

The group key must be shared only with legitimate members, and those who are no longer legitimate members or are yet to join the group should not be allowed to access the group data. To achieve this, the following requirements must be met[11][5].

- GKS (Group Key Secrecy): guarantees that it is computationally infeasible for an adversary to discover any group key.
- FS (Forward Secrecy): guarantees that a passive adversary who knows a contiguous subset of old group keys cannot discover subsequent group keys.
- BS (Backward Secrecy): guarantees that a passive adversary who knows a subset of group keys cannot discover preceding group keys.

FS is ensure that a member cannot learn about new group keys after he leaves the group. BS is ensure that a joining member cannot learn about the previous group keys even after he joins. To fulfill these requirements, KDC must update the group key whenever a member joins or leaves the group. Group members are required to buffer any encrypted data and rekey messages they received until the encrypting keys arrive[18][14][19][15]. The legal members must be guaranteed that they can decrypt the data even if the data are transmitted while they are logged off. Our objective is not to recover all of the lost keys, but to recover the only the keys that are needed to decrypt the data and useful keys. Therefore, this paper defines the condition on key recovery for reliable group key management.

- KR (Key Recovery): legitimate members must be able to recover the lost group keys and useful auxiliary keys from KDC.

3 Related Studies

There are efficient solutions for scalable group key management in dynamic groups. Fiat and Noar suggested a solution that prevents the coalitions of more than k users from group data[3]. Mittra proposed Iolus[7] in which the group

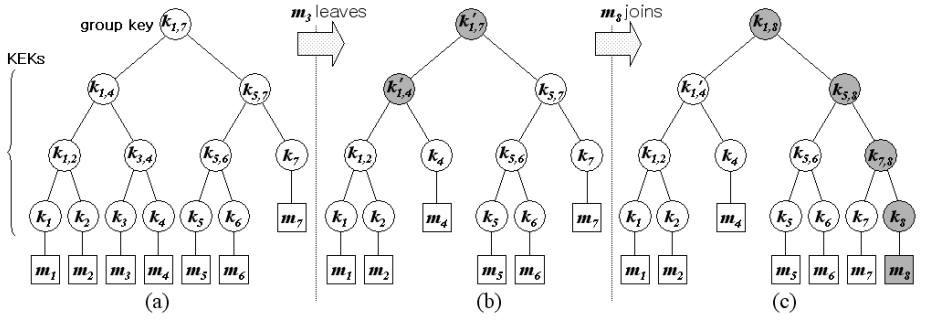

Fig. 1. A binary key-tree and key updates

is divided into several subgroups to be controlled independently. Another popular scheme, on which ours is based, is to employ a key-tree (LKH: Logical Key Hierarchy)[16][17]. LKH is secure against any number of corrupt users, and their rekey sizes are proportional to log$n$, where n is the group size, whereas Iolus requires a communication overhead proportional to the subgroup size. The improvement of the key-tree is based on the assumption that key-trees are balanced. The balance of the key-tree can be maintained by periodically updating[8] or by using AVL-tree[12]. The key values may be generated using a pseudo random number generator[17][8][12] or derived from child nodes' keys[11][1][13][2]. Since our method can be easily applied to other key-trees, the key-tree may be assumed to be in its simplest form, as a balanced binary tree. In this section, we introduce the binary key-tree with the studies for reliability.

3.1 Key Management Using a Key-Tree

Initially, KDC constructs a balanced binary key-tree whose leaf nodes nodes correspond to initial group members; the initial group may be empty. Each node of the tree is associated with a key. All members possess keys that are on the path from their leaf nodes to the root. The root key is the group key, and other keys are auxiliary keys; they will be called KEKs (Key Encryption Keys) hereafter (Refer to figure 1 (a)).

If m_3 in figure 1 (a) leaves, KDC updates the keys that m_3 possesses; KDC eliminates $k_3, k_{3,4}$ and replaces $k_{1,4}, k_{1,7}$ with $k'_{1,4}, k'_{1,7}$ (Refer to figure 1 (b)). Each new key is encrypted using its child keys and multicasted to other members; the rekey message may be such as $\{\{k'_{1,4}\}_{k_{1,2}}, \{k'_{1,4}\}_{k_4}, \{k'_{1,7}\}_{k'_{1,4}}, \{k'_{1,7}\}_{k_{5,7}}\}$. If m_8 joins in figure 1 (b), the key-tree is changed as shown in figure 1 (c). Each updated key is encrypted using the old key and the rekey message $\{\{k_{7,8}\}_{k_7}$, $\{k_{5,8}\}_{k_{5,7}}$, $\{k_{1,8}\}_{k_{1,7}}\}$ is multicasted. $\{M\}_{key}$ represents the encrypted message M of using a key, *key*. Therefore, the size of rekey message is proportional to the height of key-tree, $O(\log n)$.

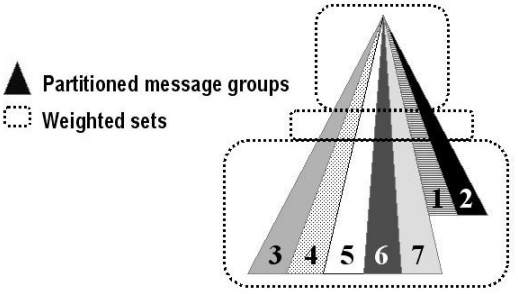

Fig. 2. Message composition

3.2 Group Key Management for Reliability

A group key distribution method with reliability was first proposed on Keystone[18]. This scheme proposes the utilization of FEC (Forward Error Correction) by the UDP transfer. [19], pointing out that the UDP transfer is incapable of dealing with burst errors, suggested a method of transferring rekey messages in divisions of many blocks. All keys needed by members are packed in a single block, and blocks are transmitted multiple times after RSE (Reed Solomon Error) codes are attached to each of them. [15] considers that keys on the upper levels of the key-tree are shared by a larger number of members, and it groups keys into several weighted classes. Keys of higher weights are entitled to more frequent transmissions. The 7 triangles in figure 2 indicate divided message blocks, and the 3 rectangles drawn in dotted lines indicate blocks with different weights. The three methods described above aim at better key update massage reception rates in soft real-time. This means that key loss cannot be completely prevented, and retransmissions are made repeatedly to prepare for key loss. However, these methods neglect the possibility of additional key updates taking place while previous retransmissions are delayed or not completed. Similarly, if a member is logged out, previous messages that should have been buffered may become lost. The two cases are the same in that the lost messages are not the last rekey messages.

In ELK[11], key verification information called *hint* is piggybacked on data for key recovery. The member who loses a rekey message message applies a brute force search for the lost keys. They can verify the candidate keys by checking them against the verification information. This method, invented to reduce data flow, necessitates large amounts of calculations by members. Nevertheless, if a member loses both the rekey message and the data, he can neither recover the lost keys nor decrypt the subsequent rekey messages. Therefore, a key recovery mechanism via KDC is necessary, not only for the recent rekey message but also for arbitrary past keys.

4 A Basic Key Recovery Mechanism

KDC maintains a binary key-tree as described in section 3.1 in our scheme. Whenever a member joins or leaves, KDC updates keys by executing procedure 1 or 2 respectively.

1. Authenticate the joining member, m_u, and send a new key, k_u, to m_u.
2. Create a leaf node n_u and a internal node n_p,
   and associate k_u and a new key k_p to them, respectively.
3. Let n_s and k_s be the shallowest leaf node in the current key tree, T_c, and its key.
   Let the corresponding keys from parent of n_s to the root be $< k_{i_1}, k_{i_2}, ..., k_{i_h} >$.
   Replace n_s with n_p, and attach n_s and n_u to n_p as child nodes.
4. Update $< k_{i_1}, k_{i_2}, ..., k_{i_h} >$ to $< k'_{i_1}, k'_{i_2}, k'_{i_h} >$
5. Encrypt $< k_p, k'_{i_1}, k'_{i_2}, k'_{i_h} >$ with k_u and unicast it to m_u.
   Encrypt k_p with k_s and k'_{i_j} with k_{i_j} $(1 \leq j \leq h)$, and multicast it to members.

Procedure 1. Join protocol

1. Let m_u and n_u be the leaving member and the corresponding leaf node
   in T_c, respectively.
   Let n_s and n_p be the sibling and the parent node of n_u, respectively.
   Let the corresponding keys from parent of n_p to the root be $< k_{i_1}, k_{i_2}, ..., k_{i_h} >$
2. Replace n_p with n_s, and remove n_u and n_p.
3. Update $< k_{i_1}, k_{i_2}, ..., k_{i_h} >$ to $< k'_{i_1}, k'_{i_2}, ..., k'_{i_h} >$
4. Encrypt each key of $< k'_{i_1}, k'_{i_2}, ..., k'_{i_h} >$ with its child node keys,
   and multicast it to remaining members.

Procedure 2. Leave protocol

A naive solution of recovering a message which is not the last rekey message is message retransmission; KDC sets the number of key recovery-supporting messages, w, as a system parameter, and prepares enough buffers in which messages of w can be stored. The appropriate value of w should be determined according to characteristics of applications. At key recovery request by a member, buffers are scanned for the requested message. The message, if it is found, is unicasted to the member. If the buffers are not holding the message, the member is notified of recovery inability. This method is simple but incomplete in that messages not stored in buffers cannot be recovered. If some of the lost KEKs are updated again before the lost rekey message is recovered, they become useless thereafter because they can no longer be used to decrypt other keys. In other words, the simple recovering of lost rekey message leads members to decrypt KEKs for which their uses have vanished. We further discuss this inefficiency in section 4.2 in detail. This section proposes a method that can perform efficient recovery at the loss of previous rekey message as well as the last message. This method also allows members to eliminate unnecessary key decryptions.

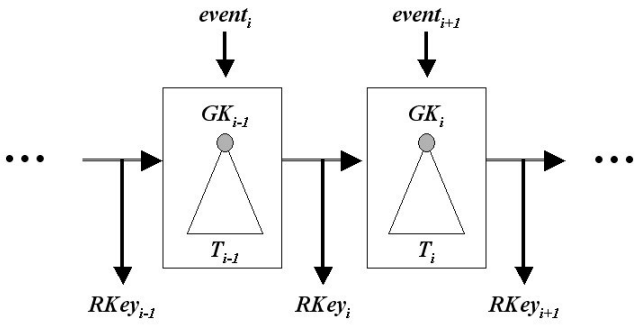

Fig. 3. Events, key-trees and rekey messages

4.1 Notations

In this section, we define the notations to explain our scheme with the following terms:

- $event_i$: An event which causes the i^{th} key update. It is either the *join* or the *leave* of a member.
- T_i: The key-tree after the i^{th} update from $event_i$ takes place (Refer to figure 3).
- $RKey_i$: The rekey message that contains keys updated from T_{i-1} to T_i (Refer to figure 3).
- GK_i: The root key of T_i which is the group key updated by $RKey_i$ in which case i is the version number of the group key (Refer to figure 1 and 3).
- m_i: The member whose identifier is i.
- k_i: The key shared only by the member m_i and KDC.
- $k_{i,j}$: The key shared by KDC and the members ranging from m_i to m_j when the key-tree is traversed by DFS (Depth First Search) (Refer to figure 1 and 3).
- $path_r^q$: The path from the leaf node of m_q to the root in T_r. For example, $path_r^1$ in figure 4 is $< k_1, k_{1,2}, k'_{1,4}, k'_{1,7} >$.
- R_r^q: The common path of $path_r^q$ and $path_r^x$ (Let the $event_r$ be caused by m_x). For example, R_r^1 in figure 4 is $< k'_{1,4}, k'_{1,7} >$.
- l_r^q: The level of the lowest node of R_r^q. For example, l_r^1 in figure 4 is the level of $k'_{1,4}$, 2 (Note that the root's level is 1 and level of a node is larger than that of the parent node by 1).
- $PRF_{key}(s)$: A pseudo random function that performs the mapping of s with the key, key[4].

This section refers to the member who has requested a key recovery as m_q, and to the lost rekey message and the group key as $RKey_r$r and GK_r, respectively. The group key, at the time of KDC receiving the key recovery request message from m_q, is GK_c.

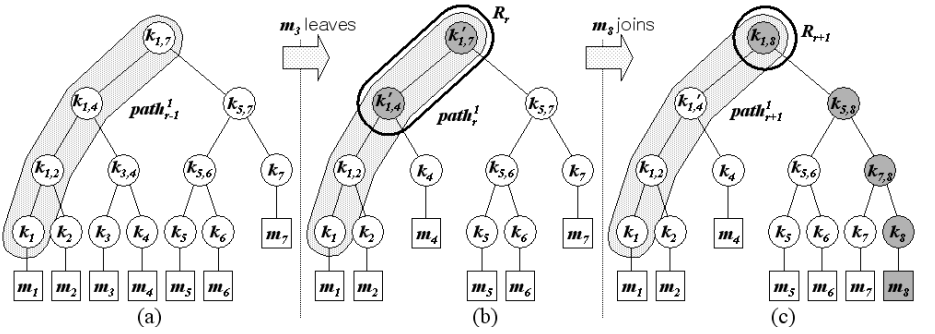

Fig. 4. Key Updates, T_i, $path_i^q$ and R_i^q

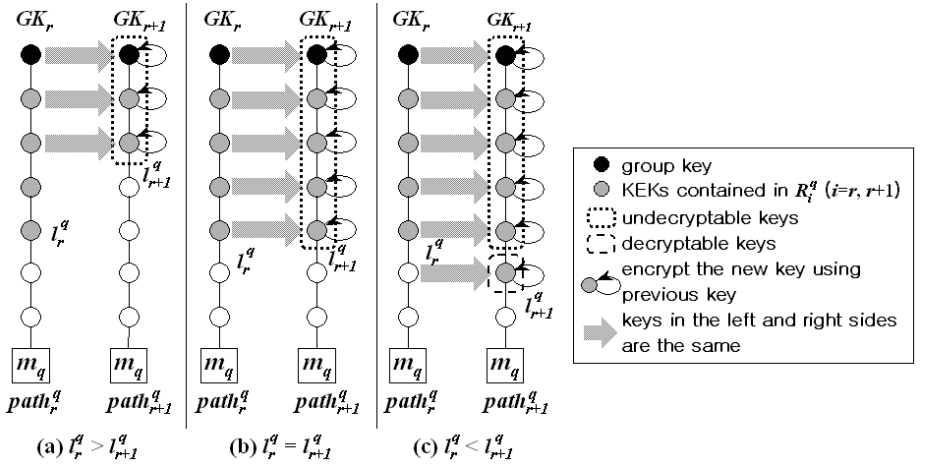

Fig. 5. In case of $event_{r+1} = join$

4.2 Observations

The following are observed from the analysis of the characteristics of LKH.

(1) If one key of the key-tree is updated due to $event_i$, then its ancestor keys are updated as well. That is, for arbitrary r and q, if we let the length of R_r^q be y, then R_r^q consists of the highest y keys of $path_r^q$ (Refer to section 4.1 and figure 4).

(2) Suppose m_q misses $RKey_r$. Then m_q cannot decrypt the following keys of $RKey_{r+1}$ due to the loss of $RKey_r$:

(a) In the case of $event_{r+1} = join$, m_q cannot decrypt the keys whose levels are less than or equal to l_r^q, since the keys used to encrypt those new keys are lost in $RKey_r$ (Refer to figure 5).
(b) In the case of $event_{r+1} = leave$ and $l_r^q > l_{r+1}^q$, m_q cannot decrypt any of the keys in R_{r+1}^q. The rekey message for R_{r+1}^q is constructed as a chain in

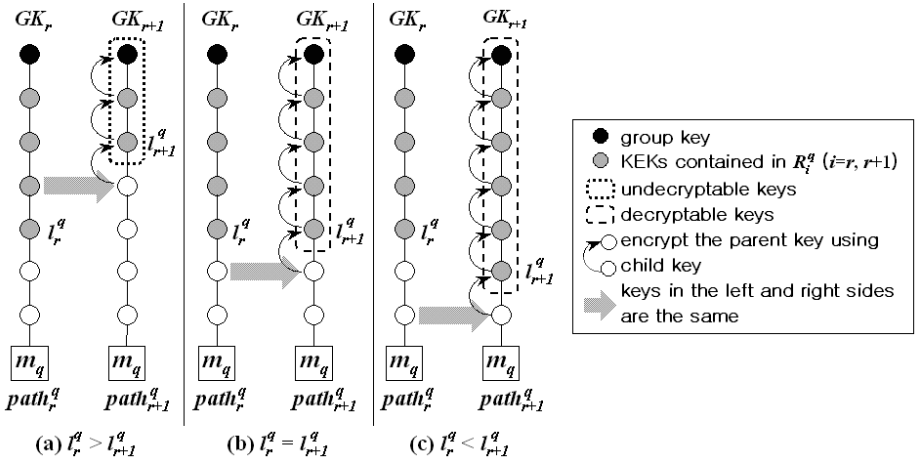

Fig. 6. In case of $event_{r+1} = leave$

which each key is encrypted using its child key. If $l^q_r > l^q_{r+1}$, the first key of the chain is lost in $RKey_r$ (Refer to figure 6 (a)). As the decryption of the chain must begin with the first key, anyone who does not know the first key cannot decrypt the keys in the chain.

(3) m_q can decrypt the following keys of $RKey_{r+1}$ without recovering $RKey_r$:

(a) In case of $event_{r+1} = join$, m_q can decrypt the nodes whose levels are greater than l^q_r, because the keys used to encrypt these keys are not lost in $RKey_r$ (Refer to figure 5 (c)).
(b) In the case of $event_{r+1} = leave$ and $l^q_r \leq l^q_{r+1}$, m_q can decrypt all the keys in R^q_{r+1}, because the first key of the chain is not lost in $RKey_r$ (Refer to figure 6 (b) and (c)). Anyone who knows the first key can decrypt all the keys in the chain.

(4) The first rekey message that m_q can decrypt without recovering $RKey_r$ is $RKey_s$. $\Longleftrightarrow$ s is the minimum value such that $l^q_r \leq l^q_s \wedge event_s = leave$.

(Proof) It can be proven by induction on s.
(a) In the case that the first index of rekey message that m_q can decrypt is $s = r + 1$:

If $event_{r+1} = leave \wedge l^q_r \leq l^q_{r+1}$, then m_q can decrypt R^q_{r+1} by (3)-(b); he can do so only if $event_{r+1} = leave \wedge l^q_r \leq l^q_{r+1}$ by (2).

(b) In the case that the first index of rekey message that m_q can decrypt is $s = r + i$, $i \geq 2$:

Since m_q cannot decrypt any of the keys of R^q_{r+1}, $event_{r+1} = join \vee l^q_r > l^q_{r+1}$ by (3). Consider the case of $event_{r+1} = join$ first. If $l^q_r > l^q_{r+1}$, the

levels of the undecryptable keys in figure 5 (a) are less than or equal to l_r^q; the two lowest keys in R_r^q are the same keys in $path_{r+1}^q$. In the case of $l_r^q \leq l_{r+1}^q$, the keys whose levels are less than or equal to l_r^q are not decryptable by (2)-(a). Similarly in the case of $event_{r+1} = leave \wedge l_r^q > l_{r+1}^q$, the keys whose levels are less than or equal to l_r^q are not decryptable. That is, m_q only does not know the keys whose levels are less than or equal to l_r^q on the $path_{r+1}^q$. The effect is the same as m_q losing R_{r+1}^q such that $l_r^q = l_{r+1}^q$. So, we can apply the same idea to j $(r < j < i)$ repeatedly. Consequently, if m_q can decrypt R_i^q, then $event_i = leave \wedge l_r^q \leq l_i^q$. And as R_i^q is the first decryptable rekey message, $event_j = join \vee l_r^q > l_j^q$ for $j(r < j < i)$.

(5) If m_q can decrypt $RKey_s(r < s)$ without recovering $RKey_r$, then he can decrypt all rekey messages thereafter.

(Proof) Since m_q can decrypt $RKey_s$, $event_s = leave \wedge l_r^q \leq l_s^q$ (Refer to (4)). As we described above, since the levels of the keys that m_q cannot decrypt are less than or equal to l_r^q, there are no undecryptable keys on the levels that are greater than l_r^q. That is, m_q knows every keys on $path_s^q$, therefore, he can decrypt all rekey messages thereafter.

(6) The keys that are not used to encrypt any other group keys or KEKs are not necessarily recovered or decrypted. In other words, if m_q can decrypt R_s^q and recover $GK_i(r \leq i < s)$, the KEKs of R_i^q $(r \leq i < s)$ whose levels are less than or equal to l_r^q are not useful for m_q.

4.3 Basic Key Recovery Algorithm

The basic idea of our scheme is searching s that satisfies the conditions started in section 4.2 (4), and then transmitting $GK_r, ..., GK_{s-1}$ that are the lost or undecryptable group keys, to m_q. In effect, KDC eliminates the transmission of useless KEKs, and the member who loses the rekey message has no need to decrypt those keys. The maximum size of R_r^q is $\log_2 n$, the height of the key-tree. Therefore, when $(s - r) > \log_2 n$, retransmitting R_r^q may be more efficient than our scheme. However, the expected value of $s - r$ is very small; we will analyze this value in section 5. Our scheme is as follows.

(1) Group Key Generation Method

KDC generates group keys in the following manner that any version of the group key can be generated at any time, while satisfying GKS, FS and BS. $MKey$ is a secret key which is only known to KDC, and the version of the group key is increased by 1 whenever the group key is updated beginning from 1.

$$GK_i = PRF_{MKey}(i).$$

(2) Node Structure of Key-tree

Each node of the key-tree has an array $flag[0, ..., w\text{-}1]$ and $last$ to calculate l_i^q. 2-bit $flag[i\%w]$ of each node records $event_i$ that causes an update of its key value. $last$ indicates the last index of an event that causes an update of the corresponding key. Whenever $event_i$ occurs by m_x, $flag[i\%w]$ of the nodes on $path_i^x$ is set to $event_i$, and $last$ is set to i. Note that $flag$s are used repeatedly with period w. Therefore, whenever key update and key recovery procedure are executed, past values of $flag$s must be cleared and the value of $last$ adjusted appropriately.

(3) Key Recovery Algorithm

KDC maintains the current key-tree, T_c only. Receiving the key recovery request for $RKey_r$ from m_q, KDC processes the procedure 3.

```
1. Check if c − r + 1 ≤ w. Then,
   1.1 Calculate l_r^q.
       // This value is the lowest level of node of path_c^q
       // whose flag[i%w] is join or leave.
   1.2 By using flags, search for s that satisfies the conditions of observation (4).
   1.3 If such an s exists,
       1.3.1 Then, send GK_r, ..., GK_{s−1} to m_q.
       1.3.2 Otherwise,
             // It means that there are no rekey messages decryptable
             // by m_q without recovering RKey_r. In this case, although
             // KDC should send R_i^q, it does not have any value of past keys.
             // KDC can use the current key-tree only.
             Send GK_r, ..., GK_{c−1} and the keys of path_c^q whose levels are
             less than or equal to l_r^q.
             // As the levels of undecryptable keys due to the loss of RKey_r
             // are less than or equal to l_r^q, m_q can decrypt R_c^q and GK_c.
2. Else, i.e., c − r ≥ w, // KDC cannot calculate l_r^q.
   2.1 Send GK_r, ..., GK_{c−1} and path_c^q to m_q.
// The keys are unicasted to m_q after being encrypted using k_q.
```

Procedure 3. A basic key recovery algorithm

5 Analysis

5.1 Security and Reliability

Our rekey algorithm is the same as the previous algorithm[16] [17], except for the group key generation method. The group key is generated by PRF in our

scheme. PRF is calculated by using the secret key of KDC, $MKey$, as a key and the version number of the group key as a seed. Characteristics of the PRF[4] make it computationally infeasible to predict the value of the function or the key even if an attacker knows a set of group keys. Therefore, FS and BS can be guaranteed because even a member who possesses certain parts of the group keys is unable to calculate the preceding/succeeding keys or $MKey$. These properties can be provided using HMAC[6] as a PRF. Another security point is in the key recovery procedure. We assume that KDC maintains members' information, such as the subscription period, for membership management. Using this, KDC can check the validity of members' key recovery requests. An adversary may try to obtain group keys by guessing the group key or $MKey$, or by eavesdropping on the key recovery messages. Because of the property of PRF, an adversary cannot discover $MKey$ or group keys by guessing even though the version number of group key is public. Thus, GKS is guaranteed if the encryption algorithm is secure; note that the recovered keys are encrypted using the individual key of the member.

KDC is capable of recovering arbitrary group keys at members' requests because group keys can be calculated only from $MKey$ and the group key version in our scheme. KDC also calculates group keys and KEKs that cannot be decrypted without the recovery of the lost keys. Our method, therefore, provides reliability with successful key recovery.

5.2 Efficiency

(1) Additional Storage on KDC

Let K be the bit length of a key. In the naive solution, KDC needs additional storage of approximately $2wK \cdot \log n$ bits to buffer past rekey messages. The required storage is increased to about $2wn$ bits in our scheme since each node of the tree has to have a $flag$. However, this may be reduced optimally. The optimization will be described in section 6.

(2) Traffic Amount and the Number of Encryptions/Decryptions

The loads on KDC and members are estimated by the number of key encryptions/decryptions and the amount of traffic is measured by the number of transmitted keys. For the analysis of the average and the worst case, let $p = c - r + 1$ and $t = s - r$; p indicates that how many key updates occur after the loss of $RKey_r$; t indicates the number of undecryptable rekey messages due to the loss of $RKey_r$. Let the average size of R_i^j be u.

In the case that the lost rekey message is not within the recovery window w, our scheme still can recover the lost message that the naive solution cannot. Such recovery would require as much work load as would the worst case. Otherwise, in the naive solution, $\log_2 n$ or $2 \cdot \log_2 n$ keys are transmitted for join or leave events, respectively. Thus, the average number of keys to be transmitted is $1.5 \cdot \log_2 n$. Because m_q cannot decrypt $RKey_r, ..., RKey_c$ in the worst case, he may decrypt $\log_2 n$ keys for each $RKey_i$ after recovering the $RKey_r$. In average, he may

Table 1. The number of computations and transmissions

Cases	Methods \ Subjects (Criteria)		KDC (The number of encryptions)	Traffic (The number of keys)	Member (The number of decryptions)
$r \le c - w$	Naive solution		-	-	N.A.
	Proposed scheme		$p-1+\log_2 n$	$p-1+\log_2 n$	$p-1+\log_2 n$
$c-w<r\le c$	Worst case	Naive solution	0	$2\cdot\log_2 n$	$p\cdot\log_2 n$
		Proposed scheme	$p-1+\log_2 n$	$p-1+\log_2 n$	$p-1+\log_2 n$
	Average	Naive solution	0	$1.5\cdot\log_2 n$	$u\cdot t$
		Proposed scheme	t	t	t

decrypt u keys for each $RKey_i (r \le i < s)$. In our scheme, KDC transmits t keys, $RKey_r, ..., RKey_{s-1}$. In the worst case, when $t = p$, KDC transmits $p-1$ group keys, $GK_r, ..., GK_{c-1}$ and $\log_2 n$ keys of $path_c^q$. The number of encryptions of KDC and the number of decryptions of a member is equal to the number of transmitted keys.

Comparisons between the proposed method and the naive solution are made on Table 1. It shows that our scheme reduces the traffic amount and process load on members instead of the load on KDC, except for the traffic amount in the worst case. However, the practical efficiency of schemes should be estimated not by its performance in the worst case but by that on average. In section 6, we will derive the expected value of t.

6 Optimization and Analysis on Expectation

As described in section 5.2, the storage of KDC in our scheme is proportional to the group size. It is not scalable for large groups. To eliminate this constraint, we try to attach $flag$s only to the highest nodes of the key-tree instead of attaching it to all nodes. In this section, we derive the appropriate level of the node to which to attach $flag$ and analyze u and t for average efficiency.

6.1 Space Optimization

For simplicity, we assume that $n = 2^h$ for some h. Let the keys of $path_r^x$, ordered by from the root to the leaf, be $k_{i_1}, k_{i_2}, ..., k_{i_h}$. Assume also that $event_r$ occurs with the join or leave of m_x. Without loss of generality, let m_x be the rightmost member in the key-tree. Then as shown in figure 7, the leftmost $n/2$ members need only the root key k_{i_1}. Next $n/2^2$ members need 2 keys, k_{i_1} and k_{i_2}. The total number of key updates for group members, except for m_x, is $\sum_{i=1}^{h}(2^{-i}\cdot n\cdot i) = \left(2-\frac{\log_2 n+2}{n}\right)\cdot n$. If the probability of key losses for all members is uniform, the expected number of keys to be recovered, u, is $2-\frac{\log_2 n+2}{n} \approx 2$.

Using this fact, we can reduce the amount of additional storage needed by KDC. If we maintain $flag$s on only the nodes whose levels are less than or

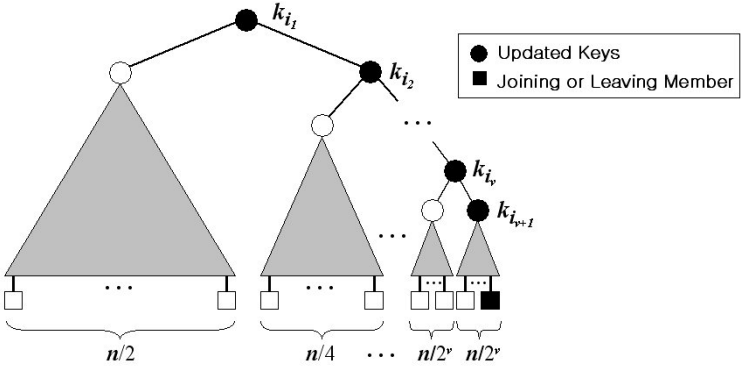

Fig. 7. The number of related members for each key

equal to v, the amount of storage is about $2w \cdot (2^v - 1)$ bits. If $l_r^q \geq v$ for some r and q, as all $flag$s of $path_r^q$ are set to $event_r$, KDC cannot know the real l_r^q. In this case, we regard l_r^q as $\log_2 h$. Therefore, the expected number of keys to be recovered converges to 2 as follows:

$$\frac{1}{n}\left\{\sum_{i=1}^{n}(2^{-i}\cdot n\cdot i)+(1-\sum_{i=1}^{n}2^{-i})\cdot n\cdot h-\frac{1}{n}\cdot h\right\}=2+\frac{\log_2 n-v-2}{2^v}-\frac{\log_2 n}{n}.$$

For various v and n, the expected number of keys to be recovered is shown in table 2. If we set v to 4 when n is not greater than 1 million, and to 5 otherwise, we can maintain the number of keys to be recovered below 3. As a result, the additional storage for KDC is reduced to $62w$ bits regardless of the group size.

6.2 Expectation of the Amounts of Traffic

Now, we analyze the expected value of t which is a major factor of efficiency in our scheme. We assume that the probabilities of the occurrence of a member join and that of a leave are equal. The probability of $l_r^q = i$ is 2^{-i}. For each i, the probabilities of $l_r^q < l_{r+1}^q$, $l_r^q = l_{r+1}^q$ and $l_r^q > l_{r+1}^q$ are 2^{-i}, 2^{-i} and $1 - 2^{-i+1}$,

Table 2. Key recovery message size, u according to v

v \ n	100	1,000	10,000	1.E+05	1.E+06	1.E+07	1.E+08	1.E+09	1.E+10
2	2.59	3.48	4.32	5.15	5.98	6.81	7.64	8.47	9.30
3	2.14	2.61	3.03	3.45	3.87	4.28	4.70	5.11	5.53
4	1.97	2.24	2.45	2.66	2.87	3.08	3.29	3.49	3.70
5	1.92	2.08	2.20	2.30	2.40	2.51	2.61	2.72	2.82
6	1.91	2.02	2.08	2.13	2.19	2.24	2.29	2.34	2.39
7	1.92	2.00	2.03	2.06	2.09	2.11	2.14	2.16	2.19
8	-	1.99	2.01	2.03	2.04	2.05	2.06	2.08	2.09
$\log_2 n$	6.64	9.97	13.29	16.61	19.93	23.25	26.58	29.90	33.22

Table 3. Expectations of computations and traffic amount

Methods \ Subjects (Criteria)	KDC (The number of encryptions)	Traffic (The number of keys)	Member (The number of decryptions)
Naive Solution	0	$1.5 \cdot \log_2 n$	6 (9)
Proposed Scheme	3	3	3

respectively. Consequently, for an arbitrary l_r^q, the probabilities of $l_r^q < l_{r+1}^q$, $l_r^q = l_{r+1}^q$ and $l_r^q > l_{r+1}^q$ are $\sum_{i=1}^{h-1}\left(2^{-2i}\right) = \frac{1}{3}\left(1 - \frac{4}{n^2}\right)$, $\sum_{i=1}^{h}\left(2^{-2i}\right) = \frac{1}{3}\left(1 - \frac{1}{n^2}\right)$ and $\sum_{i=1}^{h}\left(2^{-i} \cdot (1 - 2^{-i+1})\right) = \frac{1}{3}\left(1 - \frac{9n-6}{n^2}\right)$, respectively. Since the probability of $s = r+1$ is equal to that of $(l_r^q \leq l_{r+1}^q \wedge event_{r+1} = leave)$, it is about 1/3. The probability of $s = r+2$ is equal to that of $(l_r^q > l_{r+1}^q \vee event_{r+1} = join) \wedge (l_r^q \leq l_{r+2}^q \wedge event_{r+2} = leave)$, and it is about 2/9. In general, the probability of $s = r+t$ is about $2^{t-1}/3^t$, so the expectation of t is $\lim_{t\to\infty} \sum_{i=1}^{t}\left(t \cdot 2^{i-1} \cdot 3^{-i}\right) \cong 3$.

We implemented LKH with the balancing method proposed in [8]. The result of this experimentation shows that u converges to 3. Based on this result, the average of computations and the traffic amount are presented in table 3. As shown in table 3, the number of computation and the traffic amount of our scheme are constants, and the number of decryptions is reduced to 1/3 of the decryptions required by the naive solution in maximum.

7 Conclusion and Future Work

This paper analyzed and defined problems related to rekey message loss and to members' log-out. Based on this analysis, we proposed a key recovery mechanism that recovers arbitrary group keys efficiently and eliminates transmission and decryptions of useless KEKs without storing any rekey messages. Our scheme reduced the average traffic amount and the number of decryptions for members at the cost of encryption overheads at KDC from those of the naive solution. Our scheme is based on binary a key-tree, which is the simplest form of LKH. Other schemes such as [1] and [2] are designed to improve the efficiency of LKH. It is expected that our scheme can be applied to those schemes with slight modification. However, more exact analysis on the efficiency is necessary.

The proposed method is triggered by key recovery requests from members to KDC. Keys are more likely to be lost in cases of unstable networks. Such cases result in an increase in recovery requests, congesting data transmissions to KDC. Combinations created with methods of [18], [14] or [19] could prevent the problem. These methods alone would require repeated transmission with reception percentages close to 100%. Therefore, recovering the lost messages by the proposed method, after the application of the method of repeated transmission to guarantee appropriate degrees of reception rates, would be more efficient. Research on proper reception rates are needed in the future to maximize the efficiency.

References

1. D. Balenson, D. McGrew and A. Sherman, "Key Management for Large Dynamic Groups: One-way Function Trees and Amortized Initialization," IETF Internet Draft, 1999.
2. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient constructions," Proc. of IEEE INFOCOMM'99, Vol. 2, pp708–716, 1999.
3. A. Fiat and M. Naor, "Broadcast Encryption," Advances in Cryptology: Proc. of Crypto'93, LNCS 773, pp480–491, 1994.
4. O. Goldreich, S. Goldwasser and S. Micali, "How to Construct Random Functions," Journal of the ACM, Vol.83, Issue 4, pp792–807, 1986.
5. T. Hardjono, B. Cain and B. Doraswamy, "A Framework for Group Key Management for Multicast Security," IETF Internet Draft, 2001.
6. H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-hashing for Message Authentication," IETF RFC 2104, 1997.
7. S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," Proc. of ACM SIGCOMM'97, Vol.27, Issue 4, pp277–288, 1997.
8. M. J. Moyer, J. R. Rao and P. Rohatgi, "Maintaining Balanced Key Trees for Secure Multicast," IRTF Internet Draft, 1999.
9. http://www.ietf.org/html.charters/msec-charter.html
10. http://www.securemulticast.org/msec-index.htm
11. A. Perrig, D. Song and J. D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution," 2001 IEEE Symposium on Security and Privacy, pp247–262, 2001.
12. O. Rodeh, K. P. Birman and D. Dolev, "Optimized Group Rekey for Group Communication Systems," Network and Distributed Systems Security 2000, pp37–48, 2000.
13. S. Rafaeli, L. Mathy and D. Hutchison, "EHBT: An Efficient Protocol for Group Key Management," 3rd International Workshop on Networked Group Communications, pp159–171, 2001.
14. S. Setia, S. Zhu and S. Jajodia, "A Scalable and Reliable Key Distribution Protocol for Group Rekeying," Technical Report, George Mason Univ., 2002.
15. S. Setia, S. Zhu and S. Jajodia, "A Comparative Performance Analysis of Reliable Group Rekey Transport Protocols for Secure Multicast," Proc. of the Performance 2002 Conference, 2002.
16. C. K. Wong, M. Gouda and S. S. Lam, "Secure Group Communications using Key Graphs," Proc. of ACM SIGCOMM, Vol 28, Issue 4, pp68–79, 1988.
17. D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architectures," IETF RFC 2627, 1999.
18. C. K. Wong and S. Lam, "Keystone: A Group Key Management Service," Proc. of International Conference on Telecommunications, 2000.
19. X. B. Zhang, S. S. Lam, D. Lee and Y. R. Yang, "Protocol Design for Scalable and Reliable Group Rekeying," Proc. of SPIE Conference on Scalability and Traffic Control in IP Networks, 2001.

Efficient Software Implementation of LFSR and Boolean Function and Its Application in Nonlinear Combiner Model

Sandeepan Chowdhury and Subhamoy Maitra

Applied Statistics Unit, Indian Statistical Institute
203 B T Road, Kolkata, Pin 700 108, INDIA
sandeepan@consultant.com,
subho@isical.ac.in

.

Abstract. Here we present an efficient implementation strategy and some general design criteria for the standard nonlinear combiner model. This model combines the output sequences of several independent Linear Feedback Shift Registers (LFSRs) using a Boolean function to produce the running key sequence. The model is well studied and a standard target for many cryptanalytic attacks. The naive bitwise software implementation of the LFSRs is not efficient. In this paper we explore an efficient block oriented software implementation technique to make it competitive with the recently proposed fast stream ciphers. Our proposed specifications on this model can resist the fast correlation attacks. To evaluate our design criteria and implementation techniques, we carry out the security and performance analysis considering a specific scheme based on this model.

Keywords: Linear Feedback Shift Register, Block Oriented Software Implementation, Boolean Function, Resiliency, Nonlinearity, Algebraic Degree.

1 Introduction

Linear Feedback Shift Registers (LFSRs) are the main building block of most of the stream cipher systems. The slow software realization of the bit oriented LFSRs (i.e., LFSRs over GF(2)) reduces the efficiency of LFSR-based stream ciphers in software. To improve software implementation, recently proposed fast stream ciphers like SNOW [8,9], t-classes of SOBER [18], TURING [19] have opted for word-oriented LFSRs which are actually LFSRs over GF(2^b). The value of b is taken as 8/16/32 depending on different word sizes of the processor. However, these newly proposed fast stream ciphers are not yet time-tested and some of the newly proposed stream ciphers, like SSC2 [24], SNOW (Version-1.0), t-classes of SOBER [18], despite being very fast in software, found to have certain weaknesses in their design. In this context we like to draw attention to the well-known LFSR-based nonlinear combiner model of stream cipher. In the

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 387–402, 2003.

standard nonlinear combiner model (presented in Figure 1) the output sequences of several independent Linear Feedback Shift Registers (LFSRs) are combined using the nonlinear Boolean function to produce the running key stream K. The n LFSRs and the Boolean function are presented as $S_1, \ldots, S_n$ and f (see Figure 1) respectively. This key stream K is bitwise XORed with the message bit stream M to produce the cipher C. The decryption machinery is identical to the encryption machinery. The initial conditions of the LFSRs constitute the secret key of the system. There are several variants of this model where the

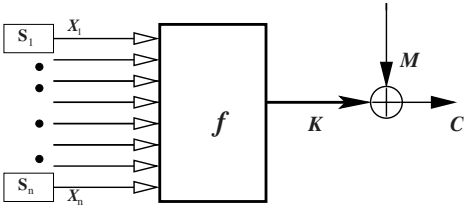

Fig. 1. Nonlinear Combiner Model.

LFSRs may be clock-controlled or the Boolean function may contain a few bits of memory. The basic nonlinear combiner model is a standard target for most of the correlation attacks [22,15,7,1,2,11,16]. As the correlation attacks on this model appear to be stabilized, it is now possible to fix the proper design criteria for the LFSRs [15,1,14] as well as that of the Boolean function [20,17,23]. So considering the present scenario, we re-examine this model in detail. At this point, let us highlight the salient features of this initiative.

1. The slow software realization of the bit oriented LFSRs reduces the efficiency of this model in software. So, for the realization of LFSRs over GF(2), we propose a block oriented technique to get considerable speed up in software. Further, we show that the algebraic normal form representation of Boolean functions can be efficiently used for block oriented strategy.
2. We start the design procedure of the basic nonlinear combiner model to resist the fast correlation attack presented by Chepyzhov, Johansson and Smeets [2]. Later, it has been checked that our design methodology provides robustness against the attack proposed by Canteaut and Trabbia [1]. We suitably incorporate recent theoretical developments in design of Boolean function and LFSRs to choose parameters of each component of the system.
3. On the basis of our analysis and implementation techniques, we present a concrete scheme, specifying the exact LFSR connection polynomials and the Boolean function. Analysing this scheme we show that it is robust against existing fast correlation attacks which performs better than the exhaustive key search. We also show that following the proposed block oriented technique, software implementation of the scheme is quite competitive with most of the other recently proposed fast software stream ciphers. However, low linear

complexity of the generated scheme still remains a problem. Some modification on the basic nonlinear combiner model may remove this weakness and make the model suitable for practical purpose.

The organization of the paper is as follows. Section 2 presents some preliminary notions. In Section 3, techniques for efficient software implementation of LFSRs and the Boolean function have been discussed. In Section 4 we develop the design approach on the basis of the existing attacks. On the basis of our design strategy, we analyse a specific scheme in Section 5.

2 Preliminaries

The basic components of the nonlinear combiner model are some LFSRs and a Boolean function.

An LFSR of length d generates a binary pseudorandom sequence following a recurrence relation over GF(2). The addition operator over $GF(2)$ is denoted by $\oplus$. It takes a small seed of length d and expands it to a much larger binary pseudorandom sequence with high periodicity. We consider a degree d primitive polynomial over GF(2) of the form $x^d \oplus \bigoplus_{i=0}^{d-1} a_i x^i$. The corresponding binary recurrence relation is $s_{j+d} = \bigoplus_{i=0}^{d-1} a_i s_{j+i}$, $j \geq 0$. Note that one uses a primitive polynomial to get maximum possible periodicity, i.e., $2^d - 1$. This polynomial is called the "connection polynomial" of the LFSR. The weight of this connection polynomial is the number of places where the coefficient has the value 1, i.e., $1 + \#\{a_i = 1\}$. The taps of the LFSR come from the positions where $a_i = 1$ and there is no tap at the positions with $a_i = 0$.

By a τ-nomial multiple of the connection polynomial, we mean a multiple of the form $x^{i_1} + x^{i_2} + \ldots + x^{i_{\tau-1}} + 1$ with degree less than $(2^d - 1)$.

An n-variable Boolean function f is defined as $f : GF(2^n) \rightarrow GF(2)$. Now we present some definitions relevant to Boolean functions.

Definition 1. *For binary strings S_1, S_2 of same length λ, we denote by $\#(S_1 = S_2)$ (respectively $\#(S_1 \neq S_2)$), the number of places where S_1 and S_2 are equal (respectively unequal). Hamming distance between S_1, S_2 is denoted by $d(S_1, S_2)$, i.e., $d(S_1, S_2) = \#(S_1 \neq S_2)$. Also the Hamming weight or simply the weight of a binary string S is the number of ones in S. This is denoted by $wt(S)$. An n-variable function f is said to be balanced if its output column in the truth table contains an equal number of 0's and 1's (i.e., $wt(f) = 2^{n-1}$).*

Definition 2. *An n-variable Boolean function $f(X_1, \ldots, X_n)$ can be considered to be a multivariate polynomial over $GF(2)$. This polynomial can be expressed as a sum of products representation of all distinct r-th order products $(0 \leq r \leq n)$ of the variables. More precisely, $f(X_1, \ldots, X_n)$ can be written as $a_0 \oplus (\bigoplus_{i=1}^{i=n} a_i X_i) \oplus (\bigoplus_{1 \leq i \neq j \leq n} a_{ij} X_i X_j) \oplus \ldots \oplus a_{12\ldots n} X_1 X_2 \ldots X_n$ where the coefficients $a_0, a_i, a_{ij}, \ldots, a_{12\ldots n} \in \{0, 1\}$. This representation of f is called the algebraic normal form (ANF) of f. The number of variables in highest order*

product term with nonzero coefficient is called the algebraic degree, or simply degree of f.

We will later show that using ANF helps faster implementation of Boolean functions when realized in block oriented manner.

Definition 3. *Functions of degree at most one are called affine functions. An affine function with constant term equal to zero is called a linear function. The set of all n-variable affine (respectively linear) functions is denoted by $A(n)$ (respectively $L(n)$). The nonlinearity of an n variable function f is $nl(f) = min_{g \in A(n)}(d(f, g))$, i.e., the distance from the set of all n-variable affine functions.*

Definition 4. *Let $\overline{X} = (X_1, \ldots, X_n)$ and $\overline{\omega} = (\omega_1, \ldots, \omega_n)$ both belong to $\{0,1\}^n$ and $\overline{X} \cdot \overline{\omega} = X_1\omega_1 \oplus \ldots \oplus X_n\omega_n$. Let $f(\overline{X})$ be a Boolean function on n variables. Then the Walsh transform of $f(\overline{X})$ is a real valued function over $\{0,1\}^n$ that can be defined as $W_f(\overline{\omega}) = \sum_{\overline{X} \in \{0,1\}^n} (-1)^{f(\overline{X}) \oplus \overline{X} \cdot \overline{\omega}}$.*

Definition 5. [10] *A function $f(X_1, \ldots, X_n)$ is m-th order correlation immune (CI) iff its Walsh transform W_f satisfies $W_f(\overline{\omega}) = 0$, for $1 \leq wt(\overline{\omega}) \leq m$. Also f is balanced iff $W_f(\overline{0}) = 0$. Balanced m-th order correlation immune functions are called m-resilient functions. Thus, a function $f(X_1, \ldots, X_n)$ is m-resilient iff its Walsh transform W_f satisfies $W_f(\overline{\omega}) = 0$, for $0 \leq wt(\overline{\omega}) \leq m$.*

By an (n, m, u, x) function we mean an n-variable, m-resilient function with degree u and nonlinearity x. It is known that for $m > \frac{n}{2} - 2$, the maximum possible nonlinearity can be $2^{n-1} - 2^{m+1}$ and such functions have three valued Walsh spectra [20]. The maximum possible algebraic degree of such functions is $n - m - 1$ [21]. Construction of such functions has been demonstrated in [23,17].

3 Software Implementation

In this section we discuss the block oriented software implementation of LFSRs and the Boolean function to enhance the processing speed.

3.1 Software Implementation of LFSRs

Hardware implementation of an LFSR generates a single bit per clock. We denote the output bit sequence (of N bits) as $\{s_j\}$, $j \leq 0 < N$. In naive software implementation, more than one (say v) logical operations (bitwise XOR, AND, and bit shifting) are required to generate a single output bit. The motivation behind the fast software implementation of an LFSR is to generate a block of b output bits in $< bv$ logical operations. For this purpose we discuss the block oriented implementation, where the LFSR outputs one block at a time. We denote this output of b bit vectors as a "block", i.e., $\mathbf{w}_j = \{s_{jb+(b-1)}, \ldots, s_{jb+1}, s_{jb}\}$. Thus

a sequence of N output bits is now obtained as N' blocks $\{\mathbf{w}_j\}, 0 \leq j < N'$, where $N = N'b$. For convenience of storage and operations in processors, it is natural to take $b = 8, 16, 32, 64$ etc.

First we discuss a block oriented implementation of an LFSR by matrix operation. The standard matrix representation of an LFSR can be seen as $\mathbf{x}_b = \mathcal{A}^b \mathbf{x}_0$, where $\mathcal{A}$ is the $d \times d$ state transition matrix and $\mathbf{x}_0$ is the d-bit initial state vector of the LFSR [12]. Note that, to get one b length block we need a binary matrix multiplication. For LFSRs of high length (say > 100), even precomputation and efficient exponentiation (for calculating $\mathcal{A}^b$) will render this technique slow.

Another approach is to extend the bitwise recurrence relation of an LFSR $s_{j+d} = \bigoplus_{i=0}^{d-1} a_i s_{j+i}$, to that over blocks $\mathbf{w}_{j+d} = \bigoplus_{i=0}^{d-1} a_i \mathbf{w}_{j+i}$. For implementing this block oriented relation, we consider an LFSR consists of d blocks corresponding to the original LFSR of d bits. We denote the LFSR blocks by $(S[d-1], \ldots, S[1], S[0])$ from left to right. The initial state is represented as $S[d-1] = \mathbf{w}_{d-1}, \ldots, S[1] = \mathbf{w}_1, S[0] = \mathbf{w}_0$. So one needs the first bd bits of the LFSR bit sequence s_j to initialize these d blocks. Once initialized, the block oriented recurrence relation can generate the output sequence of blocks i.e., $\{\mathbf{w}_j\}_{j \geq 0}$. Only $\frac{t}{b}$ logical operations are required per output bit, where t is the total number of taps. Considering a primitive trinomial (i.e., $t = 2$) and block size $b = 64$, we get $\frac{t}{b} = \frac{1}{32}$. Precomputation involving the generation of bd bits for initialization of the d blocks makes the process slow and memory dependent (≈ 3 KByte generation and storage for $d \approx 300$). Further, for synchronized operation, every reinitialization of initial d blocks is a time consuming affair. For fast implementation in software, by hard coding the LFSR involves using distinct variables for the d LFSR blocks. It is not practical when $d \approx 300$. In that case, using arrays for storing the LFSR needs indirect addressing at machine level and the implementation becomes inefficient.

We extend the idea of [24,3]. According to this approach, the LFSR (length d) consists of y number of blocks, each of length b, i.e., $d = yb$. Here we denote the blocks $(S[y-1], \ldots, S[1], S[0])$ from left to right. Initially $S[y-1] = \mathbf{w}_{y-1}, \ldots, S[0] = \mathbf{w}_0$. The block oriented recurrence relation [3] for the LFSR is

$$\mathbf{w}_{j+y} = \bigoplus_{i'=0}^{t-1} \left((\mathbf{w}_{j+q_{i'}+1} << (b - r_{i'})) \oplus (\mathbf{w}_{j+q_{i'}} >> r_{i'}) \right), \text{ for } j \geq 0. \quad (1)$$

Here $bq_{i'} + r_{i'} = p_{i'}$, and $p_0, p_1, \ldots, p_{t-1}$ are the tap positions. It may be noted that taps are only between the positions 0 and $d-b+1$. Number of logical operations required for each output bit is $\frac{t_1+4t_2}{b}$. Here, t_1 is the number of boundary taps, i.e., at any of the positions $0, b, 2b, \ldots, d(b-1)$, and t_2 is the number of non-boundary taps. Total number of taps $t = t_1 + t_2$. Considering primitive trinomials, $t_1 = 1$, $t_2 = 1$. Thus for $b = 64$, the number of logical operation per bit is $\frac{5}{64}$. Also the memory requirement is only $y = \frac{d}{b}$ blocks instead of d blocks as mentioned earlier. Further the initialization can be done directly without any precomputation. Unfortunately, the method presented in [24,3] works for LFSRs of size $d = yb$, i.e., the degree has to be multiple of block size.

Here we extend this technique for LFSR of any length $d = yb + a, 0 < a < b$, where d may not be a multiple of the block size b. For the sake of faster implementation, we consider that there is no tap between the positions $d-1$ and $(d-b-a+1)$. Here, an LFSR (denote it by M_d) of length $d = yb+a, 0 < a < b$ is assumed to consist of $y+1$ blocks, namely $(S[y], S[y-1], \ldots, S[0])$ from left to right. In order to make all the blocks of same length b, we pad the leftmost $(b-a)$ bits of the block $S[y]$ by zeroes. So at any stage we denote its state by the bit vector $\mathbf{D}_j = \{0, \ldots, 0, s_{j+d-1}, \ldots s_{j+yb}\}, j \geq 0$. Initially, $S[y] = \mathbf{D}_0, S[y-1] = \mathbf{w}_{y-1}, \ldots, S[0] = \mathbf{w}_0$. The y blocks $S[y-1], \ldots, S[0]$ can be viewed as an LFSR $(M_{d'})$ of length $d' = yb$ and the block $S[y]$ is considered separately. Recurrence relation for LFSR $(M_{d'})$ is obtained from Equation (1). The output of the LFSR (M_d) is dependent on both $M_{d'}$ and $S[y]$. Se, we need to define two simultaneous block oriented recurrence relations, for the LFSR M_d. These can be obtained as,

$$\mathbf{w}_{j+y} = (\mathbf{w'}_j << a) \oplus \mathbf{D}_{j-1} \text{ for } j \geq 0, \text{ for } M_{d'} \text{ and}$$

$$\mathbf{D}_j = \mathbf{w'}_j >> (b-a) \text{ for } j \geq 0 \text{ for single block } S[y].$$

Here, $\mathbf{w}'_j$ can be obtained from Equation (1) considering the LFSR $M_{d'}$.

Note that $\frac{t_1+4t_2+3}{b}$ logical operations are required for each output bit. Using a primitive trinomial $(t_1 = 1)$ as connection polynomial, only $\frac{8}{b}$ logical operations are required for each bit on average. The storage requirement is only $y+1$ blocks and no precomputation is required for initialization of the blocks. The implementation can be made faster by hard coding the LFSR in the program code, which involves use of only $y+1$ variables in the memory to represent the LFSRs. The pseudo code for updating the LFSR after generation of each new block will be as follows.

$$newS[0] = S[1],\ newS[1] = S[2],\ \ldots,\ newS[y-2] = S[y-1];$$

$$newS[y-1] = (newBlock << a) \oplus S[y],\ newS[y] = newBlock >> (b-a);$$

Here, $newS[\,]$ corresponds to the new state of a block $S[\,]$ and $newBlock$ corresponds to the new block obtained from LFSR $M_{d'}$. So, considering all the aspects we advocate this method for fast software implementation of an LFSR. Next we present some experimental results in Table 1.

We consider an LFSR of length 377 and calculate the number of cycles required per byte of LFSR output in actual "C" language implementation (P-IV 2.4 GHz processor, gcc compiler, LINUX (RedHat Version 8) operating system) of the technique described above. Two different block sizes of 32 and 64 have been considered. The available data type "unsigned long long" in gcc/cc compilers provides this 64 bit storage block. The corresponding theoretically calculated values of required number of logical operations per output byte (denoted as op./byte) have also been provided. For all the cases $t_1 = 1$. Note that the implementation speed is much worse than the theoretical speed and it needs some more effort to reduce the gap.

Table 1. LFSR performance for different block size.

t	$b = 64$		$b = 32$	
	cycles/byte	op./byte	cycles/byte	op./byte
2	8.04	1	8.76	2
4	14.12	2	9.29	4
6	16.80	3	9.77	6
8	20.74	4	11.08	8
10	24.14	5	12.15	10
12	28.43	6	12.70	12

3.2 Software Implementation of Boolean Function

The truth table of the Boolean function can be implemented in software using a look up table (of 2^n bits) which gives one bit output for n bit input. In order to combine the n output blocks generated by the n LFSRs, the ANF of Boolean function provides an interesting option. First we present the following construction method [23,17].

Construction 3.1 *Let f be an (n, m, d, x) function ($m > \frac{n}{2} - 2$), $f = (1 \oplus X_n)f_1 \oplus X_n f_2$, where f_1, f_2 are $(n-1)$-variable m-resilient functions. Let $F = X_{n+2} \oplus X_{n+1} \oplus f$ and $G = (1 \oplus X_{n+2} \oplus X_{n+1})f_1 \oplus (X_{n+2} \oplus X_{n+1})f_2 \oplus X_{n+2} \oplus X_n$. Also $H = (1 \oplus X_{n+3})F \oplus X_{n+3}G$. The function H constructed from f above is an $(n+3, m+2, d+1, 2^{n+1} + 4x)$ function. Moreover, F, G are both $(n+2)$-variable, $(m+2)$-resilient functions.*

Table 2. Construction of Boolean function

Algebraic Normal Form	&	$\oplus$	Total
$f = (1 \oplus X_n)f_1 \oplus X_n f_2$	2	2	4
$F = X_{n+2} \oplus X_{n+1} \oplus f$	0	2	2
$G = (1 \oplus X_{n+2} \oplus X_{n+1})f_1 \oplus (X_{n+2} \oplus X_{n+1})f_2 \oplus X_{n+2} \oplus X_n$ $= f_1 \oplus (X_{n+2} \oplus X_{n+1})(f_1 \oplus f_2) \oplus X_{n+2} \oplus X_n$	1	5	6
$H = (1 \oplus X_{n+3})F \oplus X_{n+3}G$	2	2	4
Total logical operations			16

Now consider that the algebraic normal form of f_1, f_2 are available and they need l_1, l_2 number of logical operations (AND, XOR) respectively. Following Construction 31, we require $l_1 + l_2 + 16$ logical operations to derive the function H. Table 2 gives step wise breakup of required logical operations. We consider the input variables $(X_1, \ldots, X_n)$ are n output blocks from the LFSRs. So the output of the Boolean function is one block of b bits after $l_1 + l_2 + 16$ operations. Thus b bits are produced in $l_1 + l_2 + 16$ operations. So on an average, one bit is generated in $\frac{l_1+l_2+16}{b}$ operation(s).

4 The Design Approach

Existing correlation attacks on the nonlinear combiner model exploits the correlation between the cipher text bit sequence and the bit sequence coming out of one or more LFSRs. To be specific, one considers the correlation between the sequence generated by one or more LFSRs and the cipher text C [21,22] (in turn K). Here we consider an (n, m, d, x) Boolean function f. It is known that there always exists a linear function $\bigoplus_{i=1}^{n} \omega_i X_i$ (with $wt(\omega_1, \ldots, \omega_n) = m + 1$) such that the correlation coefficient between the stream coming out of f and the linear function $\bigoplus_{i=1}^{n} \omega_i X_i$ is not zero.

Definition 6. *The sequence σ generated by XORing the sequences generated by the individual LFSRs, say, $S_{i_1}, \ldots, S_{i_{m+1}}$ is same as that produced by the LFSR S with length $L = \sum_{i=1}^{m+1} d_i$ and connection polynomial $\psi(x) = \prod_{j=1}^{m+1} c_{i_j}(x)$ [7]. We refer to this single LFSR of length L and connection polynomial $\psi(x)$ as the equivalent LFSR.*

The nonlinear combiner model can be attacked if the initial condition of an equivalent LFSR can be obtained. The standard models of correlation attacks recover the initial condition of a target LFSR from a perturbed version of the original output sequence. These attacks fit into the nonlinear combiner model, if we consider the equivalent LFSR as the target LFSR, the sequence σ as the original sequence and the running key sequence K as the perturbed sequence. Note that it is generally not possible to get K and the perturbed sequence is actually C. Naturally the attacker will consider the smallest length equivalent LFSR. So from the designer's viewpoint, fix the design parameters of the nonlinear combiner model to make the smallest length equivalent LFSR robust against correlation attacks.

The fast correlation attack proposed in [15] improved the complexity of this exhaustive search to $2^{\beta L}$ where $\beta < 1$. It has been mentioned in [15, Section 1] that the attack is not feasible if the target LFSR has more than 10 taps. So the weight of the equivalent LFSR S must be > 10. However one also needs to ensure that a high weight connection polynomial does not have a sparse multiple of low degree [15, Section 5]. This criteria has been dealt in more detail later.

The modified fast correlation attacks developed subsequently [2,11,1,16] and they transformed this problem of cryptanalysis to some decoding problem. According to this model of cryptanalysis, the running key sequence K is the perturbed version of the output sequence σ of an LFSR (here the equivalent LFSR), after passing through a Binary Symmetric Channel (BSC) with error probability $p = P[\sigma_i \neq K_i] < \frac{1}{2}$ (see Figure 2). Now we relate this error probability p of the BSC with the parameters of the nonlinear combiner model. In that case, $p = P[\sigma_i \neq K_i] = \frac{1}{2} - \frac{\max_{\omega \in \{0,1\}^n} (|W_f(\overline{\omega})|)}{2^{n+1}}$.

If we correlate the cipher text bits C with σ, then the error probability p increases further [1] and hence the complexity of the attack increases. In order to establish the robustness of our scheme we consider the correlation between σ and K. Depending on the nature of the fast correlation attacks they can be

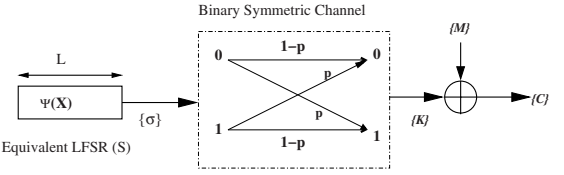

Fig. 2. Coding theory based model of Correlation Attack

divided as iterative algorithm and single pass algorithm (see [4] for details). We first consider the *single pass* algorithm described by Chepyzhov, Johansson and Smeets (CJS) [2]. The basic idea is to correctly guess a small number of output bits of the LFSR output sequence σ from the perturbed sequence C. To reduce the decoding complexity of the $[N, L]$ binary code, it is associated to another code $[n_2, k]$ of lower dimension ($k < L$). This $[n_2, k]$ code is obtained using sets of t ($t \geq 2$) parity check equations of the $[N, L]$ code [2, Algorithm A2]. This helps to recover k initial bits of the target LFSR (of length L). According to [2], the BSC has probability $p = \frac{1}{2} - \epsilon < 0.5$. The amount of cipher text bits (N) required for cryptanalysis is given by [2]

$$N = \frac{1}{4}(2k\,t!\,\ln 2)^{1/t}\epsilon^{-2}2^{\frac{L-k}{t}} \qquad (2)$$

where k is the number of initial bits recovered correctly from the attack. For given values of L and k, cipher text bit requirement N increases as ϵ decreases. Comparing the expression of p in BSC and the Boolean function model, we get $\epsilon = \frac{\max_{\omega\in\{0,1\}^n}(|W_f(\overline{\omega})|)}{2^{n+1}}$.

It is important to note here that the Boolean function needs to have highest possible nonlinearity in order to possess maximum resistance against the best affine approximation (BAA) attack [7]. Moreover, maximum nonlinearity provides the minimum value of ϵ, which in turn increases the complexity of the attack. We generally select a resilient function with three valued Walsh spectra. The (n, m, u, x) functions with best possible nonlinearity must have three valued Walsh spectra for $m > \frac{n}{2} - 2$ [20]. In this case, $\max_{\omega\in\{0,1\}^n}|W_f(\overline{\omega})| = 2^{m+2}$ and hence, $\epsilon = \frac{2^{m+2}}{2^{n+1}} = 2^{m-n+1}$. We start our design from the formulae available from [2].

4.1 Design Criteria for Equivalent LFSR

We denote the key length by q. From designer's viewpoint the time complexity of any attack should be $\geq 2^q$. The precomputation memory and time complexity of [2, Algorithm A2] are $N^{\lfloor\frac{t-1}{2}\rfloor}$ and $N^{\lceil\frac{t-1}{2}\rceil}$ [8, Section 4], where N, L, k, t are as given in Equation 2. The decoding complexity is $2^k\,k\,\frac{2\ln 2}{(2\epsilon)^{2t}}$.

Fact 4.1 *The criteria for safe design against CJS attack gives* $L = 4\,q$.

Reason 4.1 *We refer the Equation 2. From the designer's point of view we can safely approximate (underestimate) N as $2^{\frac{L-k}{t}}$. Thus, $N^{\lceil \frac{t-1}{2} \rceil}$ can be approximated as $2^{\frac{L-k}{t} \cdot \lceil \frac{t-1}{2} \rceil}$. If t is even, $\lceil \frac{t-1}{2} \rceil = \frac{t}{2}$, and $2^{\frac{L-k}{t} \cdot \lceil \frac{t-1}{2} \rceil} = 2^{\frac{L-k}{2}}$. If t is odd, then $2^{\frac{L-k}{t} \cdot \lceil \frac{t-1}{2} \rceil} = 2^{\frac{L-k}{t} \cdot \frac{t-1}{2}} = 2^{\frac{L-k}{t} \cdot \frac{t}{2}} 2^{\frac{L-k}{t} \cdot \frac{-1}{2}} = 2^{\frac{L-k}{2}} 2^{-\frac{L-k}{2t}}$. The term $2^{-\frac{L-k}{2t}}$ will be minimized when t is the smallest possible odd integer greater than 2, i.e., $t = 3$. Then the minimum value of the term is $2^{-\frac{L-k}{6}}$. So, we can underestimate $N^{\lceil \frac{t-1}{2} \rceil}$ as $2^{\frac{L-k}{2}} 2^{-\frac{L-k}{6}} = 2^{\frac{L-k}{3}}$. Hence for safe design, $\frac{L-k}{3} \geq q$, which gives $L = 3q + k$. Here, we need to keep in mind that the decoding complexity is $2^k \cdot k \frac{2 \ln 2}{(2\epsilon)^{2t}}$. Considering k as large as q, we get $2^{\frac{L-k}{3}} = 2^q$ and also the decoding complexity is greater than 2^q. Thus we take $L = 3q + k = 4q$ for a safe design.*

However, for a more tight design, we can very well decrease L by some small amount considering the other terms, mainly the terms related to ϵ.

Example 1. Let us consider $q = 256, L = 1000, \epsilon = 0.125$. Note that $4q = 1024$, but considering ϵ, we reduce L a little bit and take $L = 1000$. For this value of L, one can calculate the cipher text bit requirement, preprocessing and decoding complexity for carrying out the CJS attack for different values of t and k. Note that, in Table 3, for any given values of k and t, the corresponding maximum value of time complexity is always at least 2^{256}. Thus for $L = 1000$, the attack of [2] can succeed in no way with complexity less than the exhaustive key search, i.e., 2^{256}.

Table 3. Time complexity for CJS Attack [2] ($L = 1000, \epsilon = 0.125$)

k	t	Bit Requirement (N)	Preprocessing Complexity	Decoding complexity
		$\frac{1}{4}(2k\,t!\,\ln 2)^{1/t} \epsilon^{-2} 2^{\frac{L-k}{t}}$	$N^{\lceil \frac{t-1}{2} \rceil}$	$2^k \cdot k \frac{2 \ln 2}{(2\epsilon)^{2t}}$
32	2	2^{491}	2^{491}	2^{40}
32	3	2^{329}	2^{329}	2^{44}
64	2	2^{476}	2^{476}	2^{72}
64	3	2^{319}	2^{319}	2^{76}
96	2	2^{460}	2^{460}	2^{104}
96	3	2^{309}	2^{309}	2^{108}
128	2	2^{444}	2^{444}	2^{136}
128	3	2^{298}	2^{298}	2^{140}
160	2	2^{428}	2^{428}	2^{168}
160	3	2^{287}	2^{287}	2^{172}
192	2	2^{413}	2^{413}	2^{200}
192	3	2^{277}	2^{277}	2^{204}
224	2	2^{397}	2^{397}	2^{232}
224	3	2^{266}	2^{266}	2^{236}
256	2	2^{381}	2^{381}	2^{264}
256	2	2^{256}	2^{256}	2^{268}

Now we concentrate on the *iterative decoding* algorithm proposed by Canteaut and Trabbia (CT) [1]. The correct value of the LFSR output sequence σ are obtained by iteratively modifying the values of the sequence C using parity check equations of weight 4 and 5. According to their estimation, the cipher text bit requirement is given by $N = 2^{\alpha_\tau(p) + \frac{L}{\tau - 1}}$, where $\alpha_\tau(p) = \frac{1}{\tau - 1} log_2[(\tau - 1)! \frac{k_\tau}{C_{\tau-2}(p)}]$, $C_{\tau-2}$ is the capacity of the binary symmetric channel with overall error probability $p_{\tau-2}$, i.e., $C_{\tau-2} = p_{\tau-2} \log_2(p_{\tau-2}) + (1 - p_{\tau-2}) \log_2(1 - p_{\tau-2})$ and L is the

length of the target LFSR. Here $k_\tau \approx 3$ for $\tau = 3$ and $k_\tau = 1$ for $\tau \geq 4$. The overall error probability of the BSC is given by $p_{\tau-2} = \frac{1}{2}(1-(1-2p)^{\tau-2}$, p is the error probability of the BSC. Number of parity check equations of weight τ is $m(\tau) = \frac{N^{\tau-1}}{(\tau-1)!2^L}$. Complexity of the preprocessing stage is $\frac{N^{\tau-2}}{(\tau-2)!}$. The decoding time complexity is given by $5(\tau-1)\cdot N \cdot m(\tau)$. The total memory requirement is $2N + (\tau-1)\cdot m(d)$ computer words.

Fact 4.2 *The condition $L = 4q$ resists CT attack.*

Reason 4.2 *The time complexity of the preprocessing and decoding stages of CT attack is given by $\frac{N^{\tau-2}}{(\tau-2)!}$ and $5(\tau-1)\cdot N \cdot m(\tau)$ respectively. The memory requirement for the attack is $2N+(\tau-1)\,m(\tau)$. Here τ, (weight of the parity check equation) is 3, 4 or 5. So, clearly the complexity of the attack is proportional to the cipher text bit requirement N. Now N is given as $2^{\alpha_\tau(p)+\frac{L}{\tau-1}}$. We can underestimate N as $N \approx 2^{\frac{L}{\tau-1}}$, ($\alpha_\tau(p) > 0$). Putting $\tau = 5$, the minimum cipher text bit requirement is $N \approx 2^{\frac{L}{4}}$. Putting $L = 4q$, we get $N \approx 2^q$. Hence, for $L = 4q$, the complexity of of the CT attack is $\approx 2^q$.*

It is of interest to see how the recently published algebraic attacks [6] works on the nonlinear combiner model designed with the criteria described here.

4.2 Fixing Boolean Function and LFSR Parameters

We have already noted that $\epsilon = \frac{2^{m+2}}{2^{n+1}} = 2^{m-n+1}$. For a given value of ϵ,

1. we first need to decide about n, m and
2. once n, m are fixed, the length of each LFSR is calculated.

Now the important question is deciding n and m. First of all, we are interested about resilient functions with maximum possible nonlinearity and 3-valued Walsh spectra which needs the condition $m > \frac{n}{2} - 2$ [20]. It is known that the algebraic degree of such function is $u = n - m - 1$ [23]. Further from the viewpoint of software implementation, it is better to have lesser values of n, m, i.e., less number of LFSRs which makes the software more efficient (see Subsection 3.1 for details). Thus the requirement is to get n, m satisfying the conditions (i) $m > \frac{n}{2} - 2$, and (ii) $2^{m-n+1} \leq \epsilon$.

High algebraic degree of the Boolean function ensures high linear complexity for the output key stream K obtained from the Boolean function f [7]. For achieving high linear complexity one needs n-variable m-resilient Boolean functions with the maximum possible algebraic degree $n-m-1$ [21]. It is now known that the resilient function with maximum nonlinearity must have the maximum algebraic degree too [20].

From Definition 6, length of an equivalent LFSR $L = \sum_{j=1}^{m+1} d_{i_j}$. As the attacker will target the smallest length equivalent LFSRs, we need the average length of an LFSR (or degree of its connection polynomial) $d_{av} = \frac{L}{m+1}$.

Next we fix the degree of the LFSRs $d_1, d_2, \ldots, d_n$ and the criteria for connection polynomials. The degrees need to be pairwise coprime [12, Page 224] in order to maximize linear complexity of the running key sequence K. Also we need that $d_{i_1} + d_{i_2} + \ldots + d_{i_{m+1}} \geq L$ for any subset of $m+1$ LFSRs. Now we consider the following fact.

Fact 4.3 *The connection polynomial $\psi(x)$ of equivalent LFSR must have weight > 10, but weight of the connection polynomial $c_i(x), 1 \leq i \leq n$, of the individual LFSRs S_i should be low (< 10).*

Reason 4.3 *To resist the attacks of Meier and Stafflebach [15], the connection polynomial $\psi(x)$ of equivalent LFSR must have a high (> 10) weight (already explained in Subsection 4). The number of logical operations required for generating the output from an LFSR is dependent on the number of XOR operations (i.e., the number of taps) in its recurrence relation. So, for the sake of fast implementation in software, we need connection polynomial of individual LFSRs of low weight (see Subsection 3.1 for details). These two apparently contradictory requirements are achievable. For example, consider three trinomials of degree d_1, d_2, d_3, say. If properly chosen, the product polynomial may have weight as high as 27. Thus we have to ensure that weight of $\prod_{j=1}^{m'} c_{i_j}(x)$ should be high, for any subset (of LFSRs) having size m' $(m+1 \leq m' \leq n)$.*

Now consider about τ-nomial multiples of $\psi(x) = \prod_{j=1}^{m+1} c_{i_j}(x)$ of degree L. It has been explained in [1,14] that the expected degree of the least degree τ-nomial multiple of $\psi(x)$ is of the order of $2^{\frac{L}{\tau-1}}$. The most important attack presented using τ-nomial multiples is the CT attack [1]. In that case, the value of τ has been taken as $3, 4, 5$. Note that, even if $\tau = 5$, the minimum degree τ-nomial multiple is of the order $2^{\frac{L}{4}} = 2^q$ and the CT attack seems to be infeasible under this circumstances.

5 A Concrete Scheme

On the basis of the design criteria discussed so far, we analyse a specific scheme of the nonlinear combiner model with key length $q = 256$. We consider a block size of $b = 64$ bits. As discussed in Section 4, we start with $L = 4q = 1024$. However, considering $\epsilon = 0.125$ (i.e., $p = 0.375$), the actual value of L can be decreased to ≈ 1000. We take a $(6, 2, 3, 24)$ Boolean function. Here $n = 6$, $m = 2$ satisfies thei two conditions (i) $m > \frac{n}{2} - 2$, and (ii) $2^{m-n+1} \leq \epsilon$. We follow the Construction 31 with $f = (1 \oplus X_3)X_1 \oplus X_3X_2$, i.e., $f_1 = X_1, f_2 = X_2$. In this case no operation is required to calculate f_1, f_2 and hence $l_1 = l_2 = 0$. So, 16 logical operations are needed to get an output of 1 block from the Boolean function.

The average degree of the connection polynomials are $d_{av} = \frac{1000}{3} \approx 333$. We choose the LFSRs of length 332, 333, 337, 343, 353, 377 and take the primitive trinomials $x^{332} \oplus x^{123} \oplus 1$, $x^{333} \oplus x^{331} \oplus 1$, $x^{337} \oplus x^{135} \oplus 1$, $x^{343} \oplus x^{205} \oplus 1$,

$x^{353} \oplus x^{69} \oplus 1$, $x^{377} \oplus x^{75} \oplus 1$. Note that the degrees are mutually coprime. Considering the three ($m + 1 = 2 + 1 = 3$) least lengths out of these 6 LFSRs, $L = 1002$, which is greater than our initial design assumption (i.e., 1000). Take the initial state vector considering all the LFSRs, i.e., in total 2075 bits. Since we choose the LFSRs of degree mutually coprime, this state will return after $(2^{332} - 1)(2^{333} - 1)(2^{337} - 1)(2^{343} - 1)(2^{353} - 1)(2^{377} - 1) \approx 2^{2075}$ states.

Table 4. Minimum weight of the connection polynomial of composite LFSRs.

No. of LFSRs taken (m')	Minimum weight	Corresponding degree
3	25	1062
4	75	1395
5	209	1698
6	495	2075

Now we look at the possible minimum weights of the connection polynomial ($\psi(x)$) corresponding to the equivalent LFSRs (see Table 4). This is obtained for different subsets (size m') of the 6 LFSRs. Here $3 \leq m' \leq 6$. The least weight of the products of m' connection polynomials is 25 (> 10). Clearly the criteria given in Fact 43 is satisfied and no fast correlation attack is possible in the line of [15]. From [14], the expected degree of the least degree τ-nomial multiple for $L = 1002$ is $2^{\frac{1002}{\tau-1}}$ which is approximately $2^{500}, 2^{334}, 2^{250}$ for $\tau = 3, 4, 5$. Finding exact multiples of such high degrees seems to be very hard. Moreover, we have checked (see Table 5) that the actual complexities of the CT [1] attack for our scheme and it is always $\geq 2^{256}$. We have already discussed the robustness of this scheme against the CJS attack (see Example 1 and Table 3).

Table 5. Time/ Memory complexity for Canteaut and Trabbia Attack [1]

τ	Cipher text bits	Memory Req.	Preprocessing Comp.	Decoding Comp.
	$N = 2^{\alpha_\tau(p)+\frac{L}{\tau-1}}$	$2N + (\tau - 1) \cdot m(d)$	$\frac{N^{\tau-2}}{(\tau-2)!}$	$5(\tau - 1) \cdot N \cdot m(\tau)$
3	2^{503}	2^{504}	2^{503}	2^{512}
4	2^{337}	2^{338}	2^{673}	2^{349}
5	2^{254}	2^{255}	2^{760}	2^{271}

As mentioned earlier in Section 1, low linear complexity still remailns a problem with this model. For this specific example, using the ANF of the Boolean function, the linear complexity of the generated key stream is found to be $\approx 2^{28}$. So 2^{29} known plain text bits may be exploited to attack the scheme using Berlekamp-Massey Shift Register synthesis algorithm [13]. The concrete scheme with specific parameters are presented here to show the way of using the criteria fixed in Section 4 to evolve a nonlinear combination generator scheme safe against the existing correlation attack. We have taken trinomials (least possible weight) as connection polynomials of individual LFSRs to maximize the speed of LFSRs in software and still get the desired level of security. Depending on the security requirement (i.e., key length q), similar schemes can be designed in the same way using required number of LFSRs, their connection polynomials and the Boolean function.

5.1 Performance in Software

We study the encryption speed of our specific scheme in software, using a simple "C" language implementation. Here $b = 64$. The theoretical calculation gives that 8 logical operations are needed to get one byte output. However, in actual implementation we get much worse result as available in Table 6 and it needs further effort to optimize it in software. The storage requirement for this implementation is less than $1KB$ for storing the LFSRs. For comparison purpose we also present the encryption speed of other stream cipher schemes in Table 7. It is clear that our scheme is quite fast compared to other schemes except SNOW in Table 7.

Table 6. Key stream Generation Speed on various platforms.

Processor/RAM	Operating System	Compiler	Speed	
			Mbps	cycles/byte
PIV 1.8 GHZ/512 MB	Linux (RH 9.0)	gcc -O3	250	54
PIV 2.4 GHZ/512 MB	Linux (RH 8.0)	gcc -O3	432	42

Table 7. Performance of Different Software stream Ciphers.

Scheme	Processor/ OS	Compiler	*Speed (Mbps)
LILI-II [5]	PIII 300MHz/Win NT	Borland C++ 5.0	6
	SPARCv9/Linux	gcc	4
SSC2 [24]	Sun Ultra 1.143 MHz/Sun Solaris	gcc -O3	143
	PII 233 MHz/Linux	gcc -O3	118
	Sun SPARK2 40 MHz/Sun	gcc-O3	22
SNOW-1.0 [8]	PIII 500 MHz/Linux	gcc -O3	610
	PIII 500 MHz/Win NT	VC++	520
	PII 400 MHz/Linux	gcc -O3	380
SNOW-2.0 [9]	PIV 1.8 GHz/Linux	gcc -O3	3,610
SOBER-$t32$	Sun Ultra Sparc	-	76
$t16$ [18]	200 MHz/ Sun Solaris		44

Acknowledgment. This work has been supported partially by the ReX program, a joint activity of USENIX Association and Stichting NLnet. The authors also like to acknowledge Prof. Bimal Roy of Indian Statistical Institute for providing initial idea behind the design.

References

1. A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In *Advances in Cryptology – EUROCRYPT 2000*, LNCS Volume 1807, pages 573–588. Springer Verlag, 2000.
2. V. V. Chepyzhov, T. Johansson and B. Smeets. A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers. In *Proceedings of FSE 2000*, pages 181–195, LNCS volume 1978. Springer Verlag, 2001.
3. S. Chowdhury and S. Maitra. Efficient software implementation of Linear Feedback Shift Registers. INDOCRYPT 2001, number 2247 in Lecture Notes in Computer Science. Pages 297–307. Springer Verlag, December 2001.

4. P. Chose, A. Joux and M. Mitton. Fast Correlation Attacks: an Algorithmic Point of View. In *EUROCRYPT 2002*, pages 209–221, LNCS volume 2332. Springer Verlag, 2002.
5. A. Clark, E. Dawson, J. Fuller, J. D. Golic, H. -J. Lee, W. Millan, S. -J. Moon, L. Simpson. The LILI-II Keystream Generator. In *Information Security and Privacy – ACISP 2002*, pages 25–39, LNCS volume 2384. Springer Verlag, 2002.
6. N. T. Courtois and W. Meier. Algebraic attack on Stream Ciphers with linear feedback. In *Advances in Cryptology – EUROCRYPT 2003*, LNCS Volume 2656, pages 345–359. Springer Verlag, 2003.
7. C. Ding, G. Xiao, and W. Shan. *The Stability Theory of Stream Ciphers*. LNCS volume 561. Springer-Verlag, 1991.
8. P. Ekdahl and T. Johansson. SNOW – a New Stream Cipher. In *Proceedings of the first open NESSIE Workshop*, Heverlee, Belgium, November 13–14, 2000.
9. P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In *Selected Areas in Cryptography, SAC 2002, August 2002*, Pages 47–61, number 2595 in Lecture Notes in Computer Science, Springer Verlag, 2003.
10. X. Guo-Zhen and J. Massey. A spectral characterization of correlation immune combining functions. *IEEE Transactions on Information Theory*, 34(3):569–571, May 1988.
11. T. Johansson and F. Jonsson. Fast correlation attacks through reconstruction of linear polynomials. In *Advances in Cryptology – CRYPTO 2000*, LNCS volume 1880, pages 300–315. Springer Verlag, 2000.
12. R. Lidl and H. Niederreiter. Introduction to finite fields and their applications. Cambridge University Press, 1994.
13. J. L. Massey. Shift-register synthesis and BCH decoding. In *IEEE Transactions on Information Theory*, Vol IT-15, July 1968.
14. S. Maitra, K. C. Gupta and A. Venkateswarlu. Multiples of Primitive Polynomials and Their Products over GF(2). In *Selected Areas in Cryptography, SAC 2002, August 2002*, Pages 214–231, number 2595 in Lecture Notes in Computer Science, Springer Verlag, 2003.
15. W. Meier and O. Stafflebach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1:159–176, 1989.
16. M. J. Mihaljevic, M. P. C. Fossorier and H. Imai. Fast correlation attack algorithm with list decoding and an application. In *Proceedings of FSE 2001*, pages 196–210, LNCS 2355, Springer-Verlag, 2002.
17. E. Pasalic, S. Maitra, T. Johansson and P. Sarkar. New constructions of resilient and correlation immune Boolean functions achieving upper bounds on nonlinearity. In *Workshop on Coding and Cryptography*, Paris, January 2001. Electronic Notes in Discrete Mathematics, Volume 6, Elsevier Science, 2001.
18. G. Rose and P. Hawkes. The t-Class of SOBER Stream Ciphers. In *Proceedings of the first open NESSIE Workshop*, Heverlee, Belgium, 2000.
19. G. Rose and P. Hawkes. Turing: a fast stream cipher. In *preproceedings of the FSE 2003*, Lund, Sweden, 2003.
20. P. Sarkar and S. Maitra. Nonlinearity bounds and constructions of resilient Boolean functions. In *Advances in Cryptology – CRYPTO 2000*, LNCS volume 1880, pages 515–532. Springer Verlag, 2000.
21. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30(5):776–780, September 1984.

22. T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, C-34(1):81–85, January 1985.
23. Y. V. Tarannikov. On resilient Boolean functions with maximum possible nonlinearity. In *Proceedings of INDOCRYPT 2000*, LNCS volume 1977, 19–30, 2000.
24. M. Zhang, C. Carrol and A. Chan. The software oriented stream cipher SSC2. In *Proceedings of FSE 2000*, pages 31–48, LNCS volume 1978, 2001.

Efficient Distributed Signcryption Scheme as Group Signcryption*

DongJin Kwak and SangJae Moon

School of Electrical Eng. & Computer Science, Kyungpook National Univ., Korea.
neverdid@m80.knu.ac.kr, sjmoon@knu.ac.kr

Abstract. The existing distributed signcryption is designed for distributing a signcrypted message to a designated group. However, it does not provide confidentiality of sender ID and its extension to a group signcryption has certain weakness. This study solves the weakness and extends to an efficient group signcryption. In addition, the proposed distributed signcryption provides with sender ID confidentiality. The extension to a group signcryption is more efficient in computational load than the existing scheme. Moreover, the group manager doesn't need to keep and generate his group member's secret information. It can enhance the strength of security in the protocol.

Keywords: Signcryption, Group signature, Public-key Cryptography.

1 Introduction

The signcryption [Zhe1] is an asymmetric cryptographic method that can simultaneously provide message confidentiality and unforgeability with a lower computational and communicational overhead based on two shortened versions (SDSS1 and SDSS2) of DSS [DSS]. As a result, various signcryption schemes have been designed with additional properties [BD98,KM02,LM00]. Originally, signcryption could only be verified by a single designated verifier. However, a variant scheme with multiple designated verifiers was proposed by Zheng for the first time [Zhe2]. Thereafter, Mu and Varadharajan proposed the distributed signcryption using distributed encryption [MN99] in [MV00], where any party can "*signcrypt*" a message and distribute it to a designated group, and any member in the receiving group can "*unsigncrypt*" the message. However, this method does not provide confidentiality to the person who signcrypts the message, while the extension for group signcryption involves lots of additional complexities in computational load and some problems as a group signcryption.

Accordingly, the current paper presents a new distributed signcryption scheme that includes sender ID confidentiality and almost the same computa-

* This research has been supported by University IT Research Center Project.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 403–417, 2003.

tional cost as the existing scheme. Since the existing signcryption scheme requires the sender certificate to verify the signcryption in advance, it cannot provide the confidentiality of sender ID without an additional overhead. In contrast, the proposed scheme avoids this extra overhead, plus non-repudiation by trusted party is offered to open the signcryption in the case of dispute. Furthermore, the proposed scheme is extended to a group signcryption with properties of group signatures and deals with the problems related to the existing extended scheme, which is too complicated to satisfying some requirements, such as anonymity and traceability. Especially, it is serious issue that a group manager generates and keeps his member's secrete information in the existing protocol. It makes a forgery possible by a malicious group manager. Considering these kind of problems, we renovate its extended group signcryption. The proposed extended scheme also has more efficient computational load for both signcryption and unsigncryption than the existing extended scheme.

The remainder of this paper is organized as follows. The next section outlines the original signcryption and the distributed encryption scheme, then Section 3 explains the existing distributed signcryption and its extension. Section 4 presents the new schemes. Section 5 analyzes the security of the proposed schemes and compares their computational cost with those of previous schemes. Some final conclusions are given in Section 6.

2 Preliminaries

This section briefly reviews the original signcryption [Zhe1] and distributed encryption [MN99]. Throughout this paper, p denotes a large prime number, ${Z_p}^*$ a multiplicative group of order q for $q|(p-1)$, and $g \in {Z_p}^*$ a primitive element.

2.1 Signcryption

Signcryption [Zhe1] refers to a cryptographic method that involves the functions of encryption and digital signature simultaneously. The main advantage of the scheme is the savings in computational cost of encryption and signature compared to traditional signature-then-encryption. Signcryption is based on the Digital Signature Standard(DSS) with a minor modification that makes the scheme more computationally efficient. The modified DSS is referred to as SDSS of which there are two versions. The total procedure is as follows. Assume Alice signcrypts a message and Bob unsigncrypts the message. $(x_a, y_a = g^{x_a})$ and $(x_b, y_b = g^{x_b})$ are the private key and public key pairs for Alice and Bob, respectively. Where $hash(\cdot)$ denotes a strong one-way hash function, $hash_k(\cdot)$ a keyed one-way hash function with key k, and $E_k(D_k)$ a symmetric encryption(decryption).

Alice

choose $z \in_R Z_q$
compute $k = y_b^z \bmod p$
split k into k_1 and k_2
compute $r = hash_{k_2}(m)$
$s = z(r + x_a)^{-1} \bmod q$ if SDSS1
$s = z(1 + x_a \cdot r)^{-1} \bmod q$ if SDSS2
$c = E_{k_1}(m)$

$\longrightarrow (c, r, s) \longrightarrow$

Bob

$k = (y_a \cdot g^r)^{s \cdot x_b} \bmod p$ if SDSS1
$k = (y_a^r \cdot g)^{s \cdot x_b} \bmod p$ if SDSS2
split k into k_1 and k_2
compte $m = D_{k_1}(c)$
verify $r? = hash_{k_2}(m)$

2.2 Distributed Encryption

Distributed encryption [MN99] has a single public key together with one or more decryption keys. As such, its range can be extended to group. Consider a group of members with a public key, referred to as the group public key. Each member of the group has a group private key that matches with the group public key. Any member of the group can then decrypt the message using his or her group private key.

Initialization of a group. A group manager that is trusted by the members of the group is assumed. The group manager is responsible for constructing the group public key and updating group members. In order to construct a group including n members, the manager selects a set of integers, $\epsilon_i \in_R Z_q$ for $i = 1, 2, ..., n$ and computes the coefficients $\alpha_0, ..., \alpha_n \in Z_q$ of the following polynomial:

$$f(x) = \prod_{i=1}^{n} (x - \epsilon_i) = \sum_{i=0}^{n} \alpha_i x^i \tag{1}$$

This function has the following property: Define $g \in Z_p^*$ and $g_i \leftarrow g^{\alpha_i} \bmod p$ for i = 0,1,...,n, which produces

$$F(\epsilon_\ell) = \sum_{i=0}^{n} {g_i}^{{\epsilon_\ell}^i} = 1 \bmod p \tag{2}$$

This is because $F(\epsilon_\ell) = g^{f(\epsilon_\ell)}$ and $f(\epsilon_\ell) = 0$ in Z_q, where ϵ_ℓ is a element of the set $\{\epsilon_i\}$. That is an important property of the system. However, there is the weak point if the polynomial $\{g_i\}$ is used as group public key. In that case, an adversary would add another illegal ϵ_i' to the set $\{\epsilon_i\}$ by using the

$\{g_i\}$ polynomial. Thus, to avoid this weakness, the method given in [MN99] is adopted. For the given $\{\alpha_0, \alpha_1, ..., \alpha_n\}$, a new set is defined $\{\alpha'_0, \alpha'_1, ..., \alpha'_n\}$, where $\alpha'_0 = \alpha_0, \alpha'_n = \alpha_n, \alpha'_1 = ... = \alpha'_{n-1} = \sum_{i=1}^{n-1} \alpha_i$. Define $\beta_i \leftarrow g^{\alpha'_i}$and $A_\ell = \sum_{i=1, j=1, i\neq j}^{n-1} \alpha_j \epsilon_\ell^i$, then the property(Equation (2)) still holds:

$$F'(\epsilon_\ell) = g^{-A_\ell} \prod_{i=0}^{n} \beta_i{}^{\epsilon_\ell^i} = g^{-A_\ell} g^{\sum_{i=0}^{n} \alpha'_i \epsilon_\ell^i} = 1 \bmod p \tag{3}$$

In order to construct a group public key, the group manager picks a random number $\gamma \in_R Z_q$, then computes its inverse γ^{-1} and parameters $\rho_\ell \leftarrow -\gamma A_\ell \bmod q$ for member ℓ. The *group public key* is defined as an $n+2$ tuple, $\{\beta_0, ..., \beta_{n+1}\} \leftarrow \{\beta_0, ..., \beta_n, g^{\gamma^{-1}}\}$. The manager keeps γ and all $\{\alpha_i\}$ secret and gives ϵ_ℓ and ρ_ℓ to a group member ℓ who then uses ϵ_ℓ and ρ_ℓ as her *group private key pair*.

Distributed Encryption and Decryption. If Alice wants to send a message m securely to a designated group G, she picks the encryption key, k, computes $w = hash(m)$, and then encrypts m to obtain the ciphertext $c = (c_1, c_2)$ as follows:

$$c_1 \leftarrow \{a_0, ..., a_{n+1}\} \leftarrow \{g^k \beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\}, \qquad c_2 = m g^k \bmod p.$$

Let Bob be the group member and his group private key pair be (ϵ_b, ρ_b). Bob does the following:

$$c'_1 \leftarrow a_0 (\prod_{i=1}^{n} a_i^{\epsilon_b^i}) a_{n+1}^{\rho_b} = g^k (\prod_{i=0}^{n} g^{w \alpha_i \epsilon_b^i}) = g^k g^{w f(\epsilon_b)} = g^k \bmod p$$

After computing c'_1 using his group private key pair (ϵ_b, ρ_b), Bob can decrypt ciphertext $c = (c_1, c_2)$ from c_2/c'_1. Any member who has his group private key pair in G can then decrypt the ciphertext c to obtain m. Once m is obtained, Bob verifies the correctness of the encryption by checking whether $c_1 = \{g^k \beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\}$ using g^k and w.

3 Existing Distributed Signcryption

This section explains the existing distributed signcryption along with its extension, then considers their security and efficiency [MV00]. The group is constructed as described in Section 2 and some of notations are also the same as in Section 2.

3.1 Distributed Signcryption

Assume that Alice is the sender who signcrypts a message m and sends the message to a designated group. Bob is a member of the designated group and he unsigncrypts the message.

The signcryption. Alice does the follows and sends to Bob or the designated group the signcrypted message (c_1, c_2, r, s).

Choose $z \in_R Z_q$ and compute $k = g^z \mathrm{mod} p$
Split k into k_1 and k_2
Compute $r = hash_{k_2}(m)$
Compute $s = z(k \cdot r + x_a)^{-1} \mathrm{mod}\ q$ if SDSS1
$s = z(k + x_a \cdot r)^{-1} \mathrm{mod}\ q$ if SDSS2
Compute $w = hash(m)$
The encrypted message is as follows:

$$c_1 \leftarrow \{a_0, ..., a_n, a_{n+1}\} \leftarrow \begin{cases} \{g^{kr}\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\} & (SDSS1) \\ \{g^k\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\} & (SDSS2) \end{cases}$$

$$c_2 = E_{k_1}(m)$$

The unsigncryption. Bob who is one of the designated group members and has his group private key pair (ϵ_b, ρ_b) can unsigncrypt the signcrypted message by discovering the secret session key k as follows:

For SDSS1

$$k \leftarrow (y_a a_0 (\prod_{i=1}^n a_i^{\epsilon_b^i}) a_{n+1}^{\rho_b})^s = (y_a g^{rk} \prod_{i=0}^n g^{w\alpha_i \epsilon_b^i})^s$$
$$= (y_a g^{rk} g^{wf(\epsilon_b)})^s = g^z \mathrm{mod}\ p$$

For SDSS2

$$k \leftarrow (y_a^r a_0 (\prod_{i=1}^n a_i^{\epsilon_b^i}) a_{n+1}^{\rho_b})^s = (y_a^r g^k \prod_{i=0}^n g^{w\alpha_i \epsilon_b^i})^s$$
$$= (y_a^r g^k g^{wf(\epsilon_b)})^s = g^z \mathrm{mod}\ p$$

Then Bob splits k into k_1 and k_2 as agreed earlier and verifies $m? = D_{k_1}(c_2)$

Its extension to a group signcryption. The distributed signcryption can be extended to a group signcryption, which enables a member of group to signcrypt a message on behalf of the group and send it to another member in another group. Consider two designated group G_A and G_B, then assume that Alice belongs to the group G_A and wants to send a signcrypted message m to the group G_B which Bob belongs to. Let's introduce Alice's ϵ_a that is one of his group private key pair satisfying $f(\epsilon_a) = 0 \mod p$ (defined in section 2.2). Bob also has a counterpart key ϵ_b. The group public keys for G_A and G_B are $\{\overline{\beta}_0, \overline{\beta}_1, ..., \overline{\beta}_{n+1}\}$ and $\{\beta_0, \beta_1, ..., \beta_{n+1}\}$, respectively. Both public keys have the same form as those given earlier. In order to signcrypt the message, Alice needs to do the following and keeps (z, k, w) secret:

Choose $z \in_R Z_q$ and compute $k = g^z \bmod p$
Split k into k_1 and k_2
Compute $w = hash(m)$
Compute the signing commitment, $u_j \leftarrow \overline{\beta}_j{}^w$ for $j = 1, ..., n$
Compute $r = hash_{k_2}(m)$
Compute $s_j = z(\epsilon_a^j - ru_j) \bmod q$ for $j = 1, ..., n$

The signcrypted message is as follows:

$c_1 \leftarrow \{a_0, ..., a_n, a_{n+1}\} \leftarrow \{g^{kr}\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\}$
$c_2 \leftarrow \{\overline{a}_0, u_1, ..., u_n, \overline{a}_{n+1}\} \leftarrow \{g^{z-rk}\overline{\beta}_0{}^w, u_1, ..., u_n, \overline{\beta}_{n+1}{}^{w\rho_a}\}$
$c_3 = E_{k_1}(m)$

Sends the signcrypted tuple $(c_1, c_2, c_3, r, s_1, ..., s_n)$ to G_B.

Then The verification process by Bob belonging to G_B includes session key recovery as in the following step:

Recover session key k

$$\begin{aligned} k &= [a_0(\textstyle\prod_{i=1}^n a_i^{\epsilon_b^i})a_{n+1}^{\rho_b}][\overline{a}_0(\prod_{j=1}^n u_j^{ru_j}\overline{\beta}_j{}^{s_j})\overline{a}_{n+1}] \\ &= [g^{kr}\beta_0^w(\textstyle\prod_{i=1}^n \beta_i^{w\epsilon_b^i})\beta_{n+1}^{\rho_b}][g^{z-rk}\overline{\beta}_0^w(\prod_{j=1}^n \overline{\beta}_j{}^{w\epsilon_a^j})\overline{\beta}_{n+1}^{w\rho_a}] \\ &= [g^{rk}\textstyle\prod_{i=0}^n \beta_i^{w\epsilon_b^i}][g^{z-rk}\prod_{j=0}^n \overline{\beta}_j{}^{w\epsilon_a^j}] \\ &= g^z \bmod p \end{aligned}$$

Then Bob verifies the correctness of a_0 and $\overline{a}_0$ using the recovered session key.

3.2 Consideration

Distributed signcryption and its extension can be useful in a distributed environment where an entity signcrypts a message and a group unsigncrypts the message. However, it can not provide sender ID confidentiality and its extension is unsuitable for a group signcryption. Thus, the current study mainly focuses on dealing with these two problems.

Distributed signcryption. Distributed signcryption cannot provide sender ID confidentiality, as in the unsigncrypting procedure, the verifier cannot verify the signcryption message (c_1, c_2, r, s) without the signer's public key y_a as follows:

For SDSS1	For SDSS2
$k \leftarrow (y_a a_0(\prod_{i=1}^n a_i^{\epsilon_b^i})a_{n+1}^{\rho_b})^s$	$k \leftarrow (y_a^r a_0(\prod_{i=1}^n a_i^{\epsilon_b^i})a_{n+1}^{\rho_b})^s$

In this case, either the sender must give his certificate to the designated group in advance or someone who belongs to the designated group and wants to verify the signcryption must know the sender's public key, y_a based on their own efforts. This, of course, is another overhead in this protocol. Therefore, distributed signcryption cannot be used in applications requiring sender ID confidentiality such as electronic voting scheme and registration step in group signature. In electronic voting, sender information should be hided to other voters except administrators and in group signature, anyone who wants to join a group registers his commitment that represents himself to group manager without revealing his secret information.

Its unsuitable extension to group signcryption. Group signcryption should include the properties of both group signature and signcryption. The following outlines the weakness as regard group signcryption.

First, when initially constructing a group, the group manager computes and creates group private key pairs $\{(\epsilon_i, \rho_i)\}_{i=1}^{n}$ for all the members. In this case, if an adversary successfully attacks the manager, all secret information about the group members will be compromised. For example, assume that a member's group private key $(\epsilon_\ell, \rho_\ell)$ is revealed after an attack on the group manager. The attacker can impersonate the member and forge a signcryption message. As such, this kind of situation is very dangerous in the protocol related to groups. Furthermore, a malicious group manager can forge one of his group member easily. Because the above reasons, most of the group signatures avoid revealing the member's secret information to the manager [AT00,BS01,CS97,LW02].

Second, in unsigncryption, the verifier needs to know which group member sent the signcrypted message in advance, thereby it creates more overhead in the signcryption protocol, where all messages are encrypted by symmetric and asymmetric encryption.

Finally, the overall protocol is too complicated as regards the modular arithmetic to satisfying some requirements such as anonymity and traceability.

4 The Proposed Schemes

This section presents a new distributed signcryption scheme with sender ID confidentiality and extends it to an efficient distributed signcryption as a group signcryption by solving the above mentioned weakness.

4.1 A Distributed Signcryption with Sender ID Confidentiality

The group manager initializes a group, as in Section 2.2, then remainder of the situation is that same as in the existing scheme.

Signcryption. Alice does the following and sends to Bob who belongs to a designated group the message (c_1, c_2). Where, $Cert_a$ is Alice's certificate including

her public key, $x||y$ means the concatenation of x and y and the rest of notations are the same as in Section 3.

Choose $z \in_R Z_q$ and compute $k = g^z \text{mod} p$
Split k into k_1 and k_2
Compute $r = hash_{k_2}(m)$
Compute $s = z(r + x_a)^{-1} \text{mod } q$ if SDSS1
$s = z(1 + x_a \cdot r)^{-1} \text{mod } q$ if SDSS2
Compute $w = hash(m)$
The encrypted message is as follows:
$c_1 \leftarrow \{a_0, ..., a_n, a_{n+1}\} \leftarrow \{k\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\}$
$c_2 = E_{k_1}(m||r||s||Cert_a)$

Instead of using the session key, k, for computing s, it is used in computing c_1 in the form of $k\beta_0^w$ like ElGamal encryption [ElGa]. This method make it possible to recover the session k without knowing r, s and insert r, s, and $Cert_a$ in c_2 encryption. Therefore, the property of sender ID confidentiality is provided in the proposed scheme.

The unsigncryption. Bob or any one of the designated group members can unsigncrypt the signcrypted message using his (ϵ_b, ρ_b) based on discovering the secret session key k as follows:

$k \leftarrow a_0(\prod_{i=1}^n a_i^{\epsilon_b^i}) a_{n+1}^{\rho_b} = g^z \prod_{i=0}^n g^{w\alpha_i \epsilon_b^i} = g^z g^{wf(\epsilon_b)} = g^z \text{mod } p$
Split k into k_1 and k_2
Decrypt $D_{k_1}(c_2) = m||r||s||Cert_a$
Verify $r? = hash_{k_2}(m)$
$g^z? = (y_a \cdot g^r)^s$ if SDSS1 or $g^z? = (g \cdot y_a^r)^s$ if SDSS2

Instead of using only the session key recovery in unsigncryption, signature verification follows the session key recovery. With this two step, sender ID confidentiality can be provided. It is obvious that the recipient can unsigncrypt the message by following process above. The only way to decrypt is to have one of group member's secret key pair, ϵ_b, along with ρ_b, which gives $f(\epsilon_b) = 0 \text{ mod } q$. Alice has to use the designated group public key, otherwise no one in the group can unsigncrypt the message. Alice also has to use her private key, x_a, to compute s, since his public key is used to verify the signcryption. If s doesn't embed x_a, x_a cannot be removed by using public key y_a in the verification.

4.2 An Efficient Extension to Group Signcryption

Next, the proposed scheme is extended to a efficient group signcryption. There are five procedures involved in group signcryption: setup, join, signcryption, unsigncryption, and tracing or open, which means opening the identity of the

group member who issued the signcryption together with a proof of this fact, enabling a member of one group to signcrypt a message on behalf of the group and send it to another member in another group with anonymity.

Setup. To derive the information related a group, the group manager computes the following values:

- p, q, and g are the same as the above and $h \in Z_p^*$ is newly generated.
- G_A group manager's RSA signature key denotes d_A , verification key e_A, and n_A a large RSA modulus with two random prime factors of approximately equal length. It satisfies $e_A \cdot d_A \equiv 1 \bmod \phi(n_A)$, where $\phi(n_A)$ is Euler phi function.

The manager keeps d_A as his secret signature key and opens (p, q, g, h, n_A, e_A) as the system parameters.

Join. Each entity who wants to join a group generates his own group private key ϵ_ℓ and computes $\tau_\ell (= h^{\epsilon_\ell} \bmod p)$ as *group membership key.* Then he transfers τ_ℓ to the group manager through secure channel and proves to the manager that he knows the discrete logarithm of τ to the base h. ϵ_ℓ should be kept secret by the entity. The group manager doesn't need to keep and generate his group member's all secret information. It can enhance the strength of security. Each group manager generates $v_\ell (= \tau_\ell^{d_A} \bmod n_A)$ as membership certificate like [CS97]. Using RSA signature is simple and efficient for verifying whether τ_ℓ is valid or not. In order to setup a group, the manager computes the coefficients of following polynomial:

$$f(x) = \prod_{i=1}^{n} (x - \tau_i) = \sum_{i=0}^{n} \alpha_i x^i \tag{4}$$

Let $\{\alpha_i'\}$ and $\{\beta_i\}$ be define like Section 2.2, and define $A_\ell = \sum_{i=1, j=1, i \neq j}^{n-1} \alpha_j \tau_\ell^i$, then Equation (4) has the following property:

$$F'(\tau_\ell) = g^{-A_\ell} \prod_{i=0}^{n} \beta_i{}^{\tau_\ell^i} = g^{-A_\ell} g^{\sum_{i=0}^{n} \alpha_i' \tau_\ell^i} = g^{f(\tau_\ell)} = 1 \bmod p \tag{5}$$

In order to make a group public key, the manager picks a random number γ and computes its inverse and ρ_ℓ as in the distributed encryption. The group public key is defined as $\{\beta_0, ..., \beta_{n+1}\} \leftarrow \{\beta_0, ..., \beta_n, g^{\gamma^{-1}}\}$. Then the manager keeps γ, all $\{\alpha_i\}$, and $\{\tau_\ell\}$ secret and gives v_ℓ and ρ_ℓ to group member ℓ.

Signcryption. After the above group construction, consider two designated groups, G_A and G_B , and assume that Alice belongs to G_A and Bob is one of recipients belonging to G_B. In order to signcrypt the message m, Alice needs to do the following using his ϵ_a, τ_a, and v_a :

Choose z and $t \in_R Z_q$ and compute $k = g^z \text{mod} p$
Split k into k_1 and k_2
Compute $r = hash_{k_2}(m)$
Compute $s = z(r + \epsilon_a \cdot t)^{-1} \text{mod } q$ if SDSS1
$s = z(1 + \epsilon_a \cdot r \cdot t)^{-1} \text{mod } q$ if SDSS2
Compute $w = hash(m)$
Compute $\lambda_a = (t^{e_A} \cdot \tau_a \bmod n_A) \text{mod } q$, $\delta_a = g^{\epsilon_a t}$, and $\theta_a = t \cdot v_a \bmod n_A$
The encrypted message is as follows:
$c_1 \leftarrow \{a_0, ..., a_{n+2}, \} \leftarrow \{k\beta_0^{w\tau_a}, \beta_1^{w\tau_a}, ..., \beta_{n+1}^{w\tau_a}, g^{\lambda_a}\}$
$c_2 = E_{k_1}(ID_{G_A}||m||r||s||\delta_a||\theta_a)$

Where, ID_{G_A} is identity of group G_A. It also includes its group manager's public information (n_A, e_A) . The rest notations are the same as in previous section.

Unsigncryption. Bob or any member of G_B can unsigncrypt the signcrypted message using his (τ_b, ρ_b) based on discovering the secret session key k as follows:

$k \leftarrow a_0(\prod_{i=1}^{n} a_i^{\tau_b^i}) a_{n+1}^{\rho_b} = g^z \prod_{i=0}^{n} g^{w\alpha_i \tau_b^i} = g^z g^{wf(\tau_b)} = g^z \text{mod } p$
Split k into k_1 and k_2
Decrypt $D_{k_1}(c_2) = ID_{G_A}||m||r||s||\delta_a||\theta_a$
Compute $\lambda'_a = ({\theta_a}^{e_A} \bmod n_A) \bmod q$
Verify $r? = hash_{k_2}(m)$
$g^z? = (\delta_a \cdot g^r)^s$ if SDSS1 or $g^z? = (g \cdot \delta_a^r)^s$ if SDSS2
$a_{n+2}? = g^{\lambda'_a}$

It is obvious that the recipient can unsigncrypt the signcrypted message by the above process. The only way to decrypt is to have a group membership key, τ_b, along with ρ_b, which gives $f(\tau_b) = 0 \bmod q$. Alice has to use the designated group public key, otherwise no one in the group can unsigncrypt the signcrypted message. Alice also has to use her valid group keys, ϵ_a and τ_a, and membership certificate v_a to compute the signcryption (c_1, c_2), since only valid key and certificate can be accepted in the verification of the signcryption. Even though a group manager may release the group anonymous key, τ_a or τ_b, by accident, no malicious attacker can impersonate any group member without knowing the valid pair, (ϵ_a, v_a) or (ϵ_b, v_b).

Tracing or Open. In case of disputes, Bob forwards the c_1 and $w(= hash(m))$ to group G_A's manager after decrypting c_2 message and knowing the group G_A's identity. Then, only the manager can find the group member, Alice, who issued this signcryption by testing $\{a_i \ ? = (\beta_i^w)^{\tau_\ell}\}_{i=1}^{n+1}$ for all his group members' τ_ℓ in G_A. After this procedure, disputes can be solved by the group manager. But it causes large computational loads for very large group, so there has more rooms for improving it as a future work.

5 Analysis

This section presents an analysis of the security and computational load of the proposed schemes compared with the existing schemes.

5.1 Analysis of the Signcryption with Sender ID Confidentiality

Security. For the signcryption schemes to be secure, following conditions must be satisfied [Zhe1].

- Unforgeability: A dishonest verifier is in the best position to forge signcrypted text because he knows the original message m and corresponding signature r, s. Thus, it is shown that even the dishonest verifier can not succeed to forge the signcrypted text. For successful forging attack, a dishonest verifier must find another message m' where the hash value is r or another valid signcrypted pair, $m'||r'||s'||Cert'_a$. Considering the former, it is impossible for the attacker to find other message m' with the hash value r because the keyed hash function behaves random function. Regarding the later case, even if a dishonest verifier can generate m' and its hash value r', he cannot generate corresponding s' as he doesn't know the secret key of the signer, x_a.

- Non-repudiation: When a kind of dispute occurs, the recipient forwards a decrypted signcryption pair $m||r||s||Cert_a$ to a trusted third party. Then the third party can settle the dispute by verifying the following:

  recover key $k = (y_a \cdot g^r)^s$ if SDSS1 or $k = (g \cdot y_a^r)^s$ if SDSS2
  verify $r? = hash_{k_2}(m)$

- Confidentiality: The whole signcrypted message (c_1, c_2) is encrypted by symmetric and asymmetric encryption. Therefore, it has a stronger or same confidentiality compared with [Zhe1] and [MV00]. Plus, sender ID confidentiality is also satisfied because sender's ID is included in the encrypted message, c_2. Thus, an adversary can not discover any information without decrypting the ciphertext c_2.

Computational cost. The computational cost was considered as regards modular arithmetic, including modular exponentiation, modular multiplication, and modular inverse, then compared with the existing scheme, as shown in Table 1, since most of the computational time is spent on these modular arithmetic. In Table 1, I_q is denoted as the number of inverse in mod q, $M_q(M_p)$ as the number of multiplication in mod $q(p)$, and $E_p(E_q)$ as the number of exponentiation in mod $p(q)$. As result, the proposed scheme had almost the same computational cost compared to the existing scheme, yet the proposed scheme was more efficient in signcryption. However, when the size n increased, the cost was almost the same.

Table 1. Cost comparison between the proposed scheme and the existing scheme.

Cost		Signcryption		Unsigncryption	
		SDSS1	SDSS2	SDSS1	SDSS2
The existing scheme	I_q	1	1	-	-
	M_q	3	2	-	-
	M_p	1	1	$n+2$	$n+2$
	E_q	-	-	$n-1$	$n-1$
	E_p	$n+4$	$n+4$	$n+2$	$n+3$
The proposed scheme	I_q	1	1	-	-
	M_q	1	2	-	-
	M_p	1	1	$n+2$	$n+2$
	E_q	-	-	$n-1$	$n-1$
	E_p	$n+3$	$n+3$	$n+3$	$n+3$

5.2 Analysis of the Extension to a Group Signcryption

Security. For the group signcryption schemes which include properties of signcryption and group signature at the same time to be secure, following conditions must be satisfied [AT00,BS01,CS97,LW02,Zhe1].

- Correctness: This means that the signcryption produced by a group member must be accepted by the unsigncryption, which can be shown by inspection of the protocol.

- Unforgeability: Only valid group members are able to signcrypt a message on behalf of the group, which is similar to distributed signcryption with sender ID confidentiality. $hash_k(\cdot)$, keyed hash function, behaves as a random function, while the group private key, ϵ_a, can not be revealed to anyone, making the protocol unforgeable.

- Anonymity: With a valid decrypted message, identifying the individual who signcrypt the message is computationally hard for anyone but the group manager. As the sender's information is hidden in the form of $\delta_a (= g^{\epsilon_a t})$ and $\theta_a (= t \cdot v_a)$, and t is changed at every session, no information about sender ID is revealed by $ID_{G_A}||m||r||s||\delta_a||\theta_a$ in the proposed scheme.

- Unlinkability: Deciding if two valid decrypted messages ($ID_{G_A}||m||r||s||\delta_a||\theta_a$) and ($ID_{G_A}||\hat{m}||\hat{r}||\hat{s}||\hat{\delta_a}||\hat{\theta_a}$) were computed by the same group member is computationally hard. As for anonymity, the problem of linking two decrypted message reduces to decide whether δ_a and $\hat{\delta_a}$ are released from same member or θ_a and $\hat{\theta_a}$ are released from same member. This is related to discrete logarithm and finding random number changed at every session.

- Exculpability: Neither a group member nor the group manager can signcrypt on behalf of other group members. No one can obtain any information about

a group private key ϵ_i from $\delta_i (= g^{\epsilon_i t})$. Thus, the value ϵ_i is computationally hidden. Moreover, a group manager can not signcrypt on behalf of his group member because computing discrete logarithms is assumed to be unfeasible.

– Traceability: It was shown in above protocol.

– Coalition-resistance: This means that a colluding subset of group members cannot generate a valid signcryption that the group manager is unable to link to one of the colluding group members. Each group manager signs his members' group membership key τ_i by an RSA signature and transfer it, v_i, to each member. As such, no colluding subset can generate a valid correlated ϵ_i, τ_i, and v_i in the signcryption without the help of the right member and the manager.

– Confidentiality: It is the same as the confidentiality of Section 5.1

Computational cost. The computational cost was also considered as regards modular arithmetic, as shown in Table 2. The situations and notations were almost the same as those in Table 1, M_{n_A} was the number of multiplication in mod n_A and E_{n_A} was the number of RSA exponentiation in mod n_A. In our scheme, a minor RSA signature computation was used to avoid an impersonation attack and minor inverse in mod q was used for computing s. As result, the proposed scheme was more efficient than the existing scheme. Especially, with group size n growing, our scheme was more efficient in both signcryption and unsigncryption than the existing one.

Table 2. Cost comparison between the proposed extension and the existing extension.

Cost		Signcryption		Unsigncryption	
		SDSS1	SDSS2	SDSS1	SDSS2
The existing extension	M_q	**2n+3**	**2n+3**	**n**	**n**
	M_p	**1**	**1**	**3n+3**	**3n+3**
	E_q	**n-1**	**n-1**	**n-1**	**n-1**
	E_p	**n+7**	**n+7**	**3n+1**	**3n+1**
The proposed extension	I_q	1	1	-	-
	M_q	**n+4**	**n+5**	**-**	**-**
	M_p	**1**	**1**	**n+2**	**n+2**
	E_q	**-**	**-**	**n-1**	**n-1**
	E_p	**n+5**	**n+5**	**n+4**	**n+4**
	M_{n_A}	2	2	-	-
	E_{n_A}	1	1	1	1

6 Conclusion

The current study proposed a new distributed signcryption including confidentiality of sender ID that does not involve any additional computational cost, plus an extension for efficient group signcryption. When compared with the extension of the existing scheme, the proposed scheme is more efficient in computational cost and provides additional advantages. Especially the group manager doesn't need to keep and generate his member's secret information. It can enhance the strength of security in the protocol. The new scheme has potential applications in electronic commerce and other areas.

References

[AT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Advanced in Cryptology – CRYPTO '2000*, LNCS Vol. 1880, pages 255–270, Springer Verlag, 2000.

[BD98] F. Bao and R. H. Deng, "A signcryption scheme with signature directly verifiable by public key,", in *Public Key Cryptography – PKC '98*, LNCS Vol. 1431, pages 55–59, Springer Verlag, 1998.

[BS01] E. Bresson and J. Stern, "Efficient revocation in group signatures," in *Public Key Cryptography – PKC '2001*, LNCS Vol. 1992, pages 190–206, Springer Verlag, 2001.

[CS97] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Advances in Cryptography – CRYPTO '97*, LNCS Vol. 1294, pages 410–424, Springer Verlag, 1997.

[DSS] National Institute of Standards and Technology, "Digital Signature Standard," Federal Information Processing Standards Publication FIPS PUB 186 U.S. Department of Commerce, 1994.

[ElGa] T. ElGamal, "A public-key cryptosystmes and a signature scheme based on discrete logarithms," in *IEEE Transactions on Information Theory*, Vol. IT-31(4), pages 469–472, 1985.

[KM02] D. Kwak, J. Ha, H. Lee, H. Kim, and S. Moon, "A WTLS handshake protocol with user anonymity and forward secrecy,", in *CDMA International Conference – CIC '2002*, LNCS Vol. 2524, pages 219–230, Springer Verlag, 2002.

[LM00] K. Lee and S. Moon, "AKA protocol for mobile communications,", in *Australasian Conference Informations Security and Privacy – ACISP '2000*, LNCS Vol. 1841, pages 400–411, Springer Verlag, 2000.

[LW02] Y. Lyuu and M. Wu, "Convertible Group Undeniable Signatures,", in *International Conference on Information Security and Cryptology – ICISC '2002*, LNCS Vol. 2587, pages 48–61, Springer Verlag, 2002.

[MN99] Y. Mu, V. Varadharajan, and K. Q. Nguyen, "Delegated decryption," in *Cryptography and Coding '99*, LNCS Vol. 1746, pages 258–269, Springer Verlag, 1999.

[MOV] A. J. Manezes, P. C. van Oorshot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[MV00] Y. Mu and V. Varadharajan, "Distributed signcryption," in *Advanced in Cryptology – INDOCRYPT '2000 Proceedings*, LNCS Vol. 1977, pages 155–164, Springer Verlag, 2000.

[RSA] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," in *Communications of the ACM*, Vol. 21, No.2, pages 120–126, 1978.

[Zhe1] Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption),"in *Advanced in Cryptology – CRYPTO '97 Proceedings*, LNCS Vol. 1294, pages 165–179, Springer Verlag, 1997.

[Zhe2] Y. Zheng, "Signcryption and its application in efficient public key solutions,", in *Information Security Workshop – ISW '97*, LNCS Vol. 1396, pages 291–312, Springer Verlag, 1997.

Architectural Enhancements for Montgomery Multiplication on Embedded RISC Processors

Johann Großschädl and Guy-Armand Kamendje

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria
{Johann.Groszschaedl,Guy-Armand.Kamendje}@iaik.at

Abstract. Montgomery multiplication normally spends over 90% of its execution time in inner loops executing some kind of multiply-and-add operations. The performance of these critical code sections can be greatly improved by customizing the processor's instruction set for low-level arithmetic functions. In this paper, we investigate the potential of architectural enhancements for multiple-precision Montgomery multiplication according to the so-called Finely Integrated Product Scanning (FIPS) method. We present instruction set extensions to accelerate the FIPS inner loop operation based on the availability of a multiply/accumulate (MAC) unit with a wide accumulator. Finally, we estimate the execution time of a 1024-bit Montgomery multiplication on an extended MIPS32 core and discuss the impact of the multiplier latency.

1 Introduction

An embedded system is in general designed for a pre-defined application or class of applications (i.e. an *application domain*) with typically fixed functionality. This has motivated the development of processors customized to the needs of a particular application (domain), the so-called application-specific and *domain-specific processors*. The design space that spans domain-specific processors has several degrees of freedom, including the approach to parallel processing, the elements of special-purpose hardware, the structure of memory architectures, the types of on-chip communication mechanisms, and the use of peripherals [13]. By spending silicon where it truly matters, domain-specific processors are faster and/or more energy efficient than general-purpose processors (GPPs) for the class of applications they have been designed for. Therefore, a domain-specific processor can be seen as a cross between an ASIC and a GPP, i.e. it combines the performance and efficiency of an ASIC with the flexibility and programmability of a GPP.

It is widely accepted that domain-specific processors play a significant role in the embedded systems world and will become even more important in future. However, designing a fully domain-specific processor "from scratch" is a tedious task, involving not only the hardware design effort, but also the development of supporting tools like compilers or assemblers. A less aggressive approach to

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 418–434, 2003.

achieve domain specialization is possible via processor customization based on *instruction set extensions*. That way, an existing instruction set architecture (ISA) is extended by special instructions for performance-critical operations [17]. The circuitry which actually performs a given type of operation on operands in general-purpose registers is called a *functional unit* (FU). Typical options for domain-specialization include the adding of new FUs, extending existing FUs (such as adding extra functionality to the ALU), and introducing new interfaces between FUs [12].

A steadily growing number of micro-processor vendors and intellectual property (IP) providers license configurable and extensible processor cores to their customers, e.g. ARC Tangent-A4 [1], Tensilica Xtensa [9], Hewlett-Packard & STMicroelectronics Lx platform [8], as well as MIPS Technologies Pro Series [24]. By using a common base instruction set, the design process can focus on the application-specific extensions, which significantly reduces verification effort and hence shortens the design cycle. The result of this application-specific customizations of a common base architecture are families of closely related and largely compatible processors. These families can share development tools (compilers, debuggers, simulators) and even binary compatible code which has been written for the common base architecture. Critical code portions are customized using the application-specific instruction set extensions [12,31].

In recent years, instruction set extensions for multimedia workloads have become a prominent feature in desktop computers (e.g. Intel's MMX). Instruction set extensions also provide some promising opportunities to implement public-key cryptography (PKC) in embedded systems such as smart cards. A processor with cryptography extensions offers a degree of flexibility and scalability that goes far beyond of what is possible with a cryptographic co-processor. In the context of cryptographic hardware, the term *scalability* refers to the ability to process operands of arbitrary size. An implementation of PKC on a processor with cryptography extensions is perfectly scalable since it is basically a software implementation (the only limiting factor is the available memory). Moreover, software implementations are *flexible* as they permit to use the "best" algorithm for the miscellaneous arithmetic operations involved in PKC. For instance, squaring of a long integer can be done much faster than conventional multiplication [18]. Most hardware multipliers do not implement special squaring algorithms since this would greatly complicate their architecture.

Contrary to multimedia extensions, there exist only very few research papers concerned with optimized instruction sets for PKC. Previous work [6] and [27] focussed on the ARM7 architecture. However, we selected the MIPS32 instruction set architecture [22] for our research because it is one of the most popular architectures in the embedded systems area and gathered a considerable market share in the 32-bit smart card sector [23]. In this paper, we analyze how instruction set extensions for Montgomery multiplication [25] can be implemented efficiently and what functionality is required to achieve peak performance. We applied hardware/software co-design techniques to define and evaluate the custom instructions and the FU. This is necessary since the application-specific FU

(basically a multiply/accumulate unit in our case) is directly controlled by the instruction stream. Our primary goal was to develop instruction set extensions and architectural enhancements which are simple to incorporate into common RISC architectures like the MIPS32. In particular, we aim to avoid non-trivial modifications of the processor core like the addition of extra registers or extra buses. That way, the extended processor remains fully compatible to the base architecture (MIPS32 in our case).

2 Multiple-Precision Arithmetic

A general problem in public-key cryptography is the implementation of arithmetic with high-precision operands (≥ 1024 bits) on processors with short word size (8, 16, 32, or 64 bits). In the typical case, the operand length exceeds the word size of the processor by one to two orders of magnitude. Hence, we are forced to represent these operands as multi-word data structures (i.e. *multiple-precision* numbers) and to perform arithmetic operations by means of software routines that manipulate these structures. In the following, we use uppercase letters to denote multiple-precision integers, while lowercase letters, usually indexed, represent individual words (w-bit digits). For example, an n-bit integer A can be written as an array $(a_{s-1}, \ldots, a_1, a_0)$ consisting of $s = \lceil n/w \rceil$ words (w is the processor's word size in bits, i.e. $a_i \leq 2^w - 1$).

$$A = \sum_{i=0}^{s-1} a_i \cdot 2^{i \cdot w} = a_{s-1} \cdot 2^{(s-1) \cdot w} + \cdots + a_2 \cdot 2^{2w} + a_1 \cdot 2^w + a_0 \qquad (1)$$

2.1 Multiple-Precision Multiplication

The classical algorithm for multiple-precision multiplication is the well-known *pencil-and-paper method* which requires s^2 single-precision multiplications to form the product of two s-word integers [14]. In basic terms, each word a_j of the multiplicand A is multiplied by each word b_i of the multiplier B, and the partial products $a_j \cdot b_i$ are summed up with respect to their weight $2^{(i+j) \cdot w}$.

$$A \cdot B = A \cdot \sum_{i=0}^{s-1} b_i \cdot 2^{i \cdot w} = \sum_{i=0}^{s-1} A \cdot b_i \cdot 2^{i \cdot w} = \sum_{i=0}^{s-1} \sum_{j=0}^{s-1} a_j \cdot b_i \cdot 2^{(i+j) \cdot w} \qquad (2)$$

The algorithm for computing $A \cdot B$ consists of an outer loop and a relatively simple inner loop in which a calculation of the form $a \times b + c + d$ is carried out [18]. Given single-precision (w-bit) words a, b, c, d, the result of $a \times b + c + d$ can be stored in two registers (i.e. it occupies at most $2w$ bits) because

$$a \times b + c + d \leq (2^w - 1) \times (2^w - 1) + (2^w - 1) + (2^w - 1) \leq 2^{2w} - 1 \qquad (3)$$

Each iteration of the inner loop performs a $(w \times w)$-bit multiplication and two additions along with a couple of memory accesses. Note that adding a single-

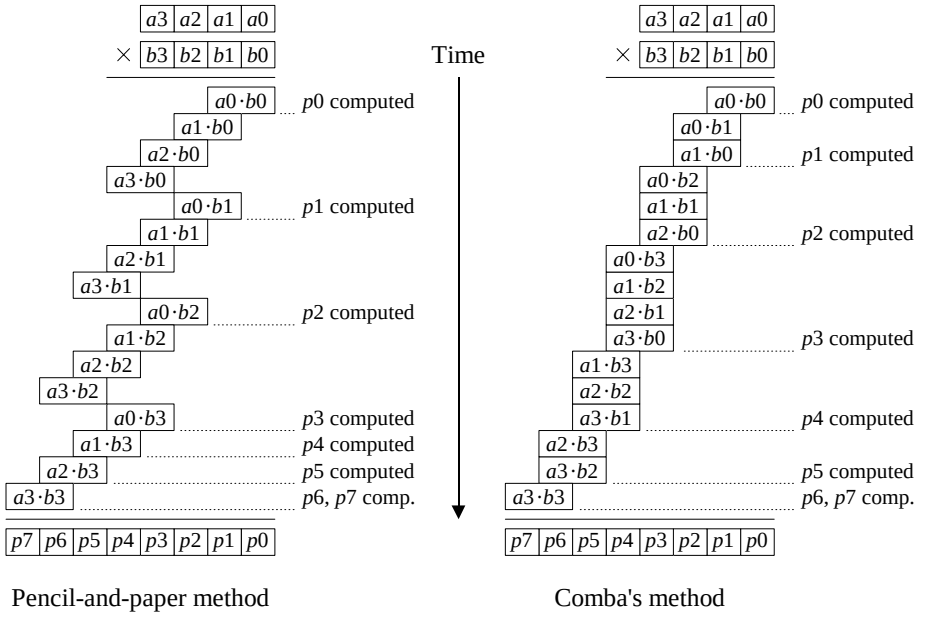

Fig. 1. Comparison of pencil-and-paper method and Comba's method (Source: [10])

precision word to a two-word product actually involves two `ADD` instructions[1] since a single-precision addition may produce a carry which has to be processed properly.

A slightly different approach for fast multiple-precision multiplication, known as *Comba's method* [5], aims to achieve better performance by reducing the number of memory store operations. This method is based on the recognition that multiplication is essentially a *convolution* of the words of the multiplicand and multiplier, followed by a carry propagation between the convolution sums to obtain the final result in radix-2^w representation. The partial products $a_j \cdot b_i$ are accumulated on a "column-by-column" basis (instead of the "row-by-row" approach used by the pencil-and-paper method). That way, the product $A \cdot B$ is formed by computing each word of the result at a time, starting with the least significant word. The main difference to the standard algorithm is the order of partial product generation/accumulation and that the carry propagation is deferred to the outer loop. However, the number of single-precision multiplications is the same, namely s^2. Comba's method minimizes the number of memory accesses by only writing a word of the result to memory when it has been completely evaluated [10] (see Figure 1).

A straightforward implementation of Comba's method ends up in a nested loop structure. The operation carried out in the inner loop is essentially *multiply-and-accumulate* $a \times b + S$ — two operands are multiplied and the product is

[1] More precisely, an `ADD` and an `ADDC` (add with carry) instruction are required.

added to a cumulative sum S. In general, a sum of k double-precision products requires $2w + \lceil \log_2(k) \rceil$ bits of storage to avoid overflow or loss of precision. An Assembly-language implementation can cope with this extra precision of S by simply accumulating the partial products into three registers.

On most architectures, Comba's method is likely to perform better than the pencil-and-paper method, especially when the inner loops are coded in Assembly language. Comba's method is generally used to implement long integer multiplication on processors with a multiply/accumulate (MAC) unit [3,7]. Most digital signal processors (DSPs) feature a MAC unit with a "wide" accumulator so that a certain number of products can be accumulated without loss of precision. For instance, Motorola's 56k-series of signal processors incorporates a $(24 \times 24 + 56)$-bit MAC unit, i.e. the accumulator register provides eight extra bits (the so-called "guard bits") for overflow protection.

2.2 Multiple-Precision Squaring

Squaring allows some specific optimizations by exploiting the symmetry in the multiplication of two identical operands. The partial products $a_j \cdot a_i$ appear once when $i = j$, and twice in the case of $i \neq j$. However, all terms of the form $a_j \cdot a_i$ and $a_i \cdot a_j$ are the same and need to be computed only once and then left-shifted in order to be doubled, i.e. $a_j \cdot a_i + a_i \cdot a_j = 2 \cdot a_j \cdot a_i$.

$$A \cdot A = \sum_{i=0}^{s-1} \sum_{j=0}^{s-1} a_j \cdot a_i \cdot 2^{(i+j) \cdot w} = \sum_{i=0}^{s-1} a_i^2 \cdot 2^{i \cdot w} + 2 \sum_{i=0}^{s-1} \sum_{j=i+1}^{s-1} a_j \cdot a_i \cdot 2^{(i+j) \cdot w} \quad (4)$$

An inspection of the above equation reveals that an s-word squaring operation requires only $(s^2 + s)/2$ single-precision multiplications (assuming that the multiplication by 2 is accomplished with `ADDC` instructions). Squaring is therefore almost twice as fast as conventional multiplication.

2.3 Montgomery Multiplication

One of the most efficient techniques for computing a modular multiplication $A \cdot B \bmod N$ was published by Peter Montgomery in 1985 [25]. Montgomery's algorithm uses a non-conventional representation of the residue classes modulo N and replaces the division by N with an addition of a multiple of N followed by a shift operation. Every integer $A < N$ is represented by its so-called N-residue (or Montgomery image) defined as $\bar{A} = A \cdot R \bmod N$. The factor R is generally selected as the least power of 2 greater than N, i.e. $R = 2^n$. There is clearly a one-to-one relationship between the the N-residues and the "original" integers in the range $[0, N-1]$. The key advantage of N-residue representation is that it allows very fast computation of the *Montgomery product*, which is defined as the N-residue of the product of two integers whose N-residues are given.

$$\bar{P} = \mathrm{MonPro}(\bar{A}, \bar{B}) = \bar{A} \cdot \bar{B} \cdot 2^{-n} \bmod N \quad (5)$$

Algorithm 1. Montgomery multiplication (FIPS method)

Input: n-bit modulus N, $2^{n-1} \leq N < 2^n$, operands $A, B < N$, $n'_0 = -n_0^{-1} \bmod 2^w$.
Output: Montgomery product $P = A \cdot B \cdot 2^{-n} \bmod N$.

$(t, u, v) \leftarrow 0$
for i from 0 by 1 to $s-1$ **do**
  for j from 0 by 1 to $i-1$ **do**
    $(t, u, v) \leftarrow (t, u, v) + a_j \cdot b_{i-j}$
    $(t, u, v) \leftarrow (t, u, v) + p_j \cdot n_{i-j}$
  end for
  $(t, u, v) \leftarrow (t, u, v) + a_i \cdot b_0$
  $p_i \leftarrow v \cdot n'_0 \bmod 2^w$
  $(t, u, v) \leftarrow (t, u, v) + p_i \cdot n_0$
  $v \leftarrow u,\ u \leftarrow t,\ t \leftarrow 0$
end for
for i from s by 1 to $2s-1$ **do**
  for j from $i-s+1$ by 1 to $s-1$ **do**
    $(t, u, v) \leftarrow (t, u, v) + a_j \cdot b_{i-j}$
    $(t, u, v) \leftarrow (t, u, v) + p_j \cdot n_{i-j}$
  end for
  $p_{i-s} \leftarrow v$
  $v \leftarrow u,\ u \leftarrow t,\ t \leftarrow 0$
end for
$p_s \leftarrow v$
if $P \geq N$ **then** $P \leftarrow P - N$ **end if**

Montgomery arithmetic requires pre-processing (to obtain the N-residues of the operands) and post-processing (to eliminate the constant factor 2^n). Therefore, the conversion to/from N-residues is only carried out before/after a lengthy computation like modular exponentiation. We do not deal with the underlying mathematics here since it is covered in a number of text books, e.g. [18].

The authors of [15] described various ways of implementing Montgomery multiplication in software. Roughly speaking, the algorithms for computing the Montgomery product can be categorized according to two simple criteria. The first criterion is whether multiplication and reduction are performed *separated* or *integrated*. In the separated approach, the modular reduction takes place after the product $A \cdot B$ has been completely formed. The integrated approach alternates between multiplication and reduction. Both coarse and fine integration are possible, depending on the frequency of switchings between multiplication and reduction steps. The second criterion is the principal order in which the operands are evaluated. One form is the *operand scanning*, where an outer loop moves through the words of one of the operands (similar to the pencil-and-paper method). Another form is *product scanning*, where the outer loop moves through the words of the result itself — just like Comba's method.

An efficient technique to compute the Montgomery product is the so-called *Finely Integrated Operand Scanning* (FIOS) method [15]. The FIOS method interleaves multiplication and reduction phases and performs them even in the

same inner loop. This finely integration is preferable over the coarsely integration (e.g. CIOS method [15]) in which multiplication and reduction take place in two separate inner loops. In particular, the FIOS method requires less memory accesses and minimizes the loop overhead since the increment of the loop counter and the branch instruction occur only once. The inner loop of the FIOS method carries out calculations of the form $(u, v) \leftarrow a \times b + c + d$, similar to the pencil-and-paper multiplication[2]. We refer to [11] for a detailed discussion of implementation aspects regarding the FIOS method.

The second approach for combining multiplication and reduction steps into a single inner loop is the so-called *Finely Integrated Product Scanning* (FIPS) method, which can be phrased according to Algorithm 1. This method was first described in [7] and is the standard way to realize Montgomery multiplication on a DSP. Each iteration of the inner loop executes two multiply-and-accumulate operations of the form $a \times b + S$, i.e. the products $a_j \cdot b_{i-j}$ and $p_j \cdot n_{i-j}$ are added to a cumulative sum. This cumulative sum is stored in the three single-precision words t, u, and v, whereby the triple (t, u, v) represents the integer value $t \cdot 2^{2w} + u \cdot 2^w + v$.

Other characteristics of the FIPS method are the more costly loop control (two nested loops instead of one) and the "reversed" addressing. The pointers to the current words of A and P move from less to more significant positions, whilst the pointers to the words of B and N move in opposite direction (i.e. they are decremented during the iterations of the inner loop). The operation at lines 10 and 18 of Algorithm 1 is essentially a w-bit right-shift of the cumulative sum (t, u, v) with zeroes shifted in.

3 The MIPS32 Instruction Set Architecture

The MIPS32 architecture is a superset of the previous MIPS I and MIPS II instruction set architectures and incorporates new instructions for standardized DSP operations like "multiply-and-add" (`MADD`) [22]. MIPS32 uses a load/store data model with 32 general-purpose registers (GPRs) of 32 bits each. The fixed-length, regularly encoded instruction set includes the usual arithmetic/logical instructions, load and store instructions, jump and branch instructions, as well as co-processor instructions.

The 4Km processor core is a high-performance implementation of the MIPS32 instruction set architecture [20]. Key features of the 4Km are a 5-stage pipeline with branch control, a fast multiply/divide unit (MDU) supporting single-cycle (32×16)-bit MAC operations, and up to 16 kB of instruction and data caches (modified Harvard architecture). Most instructions occupy the execute stage of the pipeline only for a single clock cycle. The 4Km is widely used in digital consumer products, cellular phones, and networking devices — application fields in which security becomes increasingly important.

[2] The tuple (u, v) denotes a double-precision ($2w$-bit) quantity with u and v representing the w most/least significant bits, i.e. $(u, v) = u \cdot 2^w + v$.

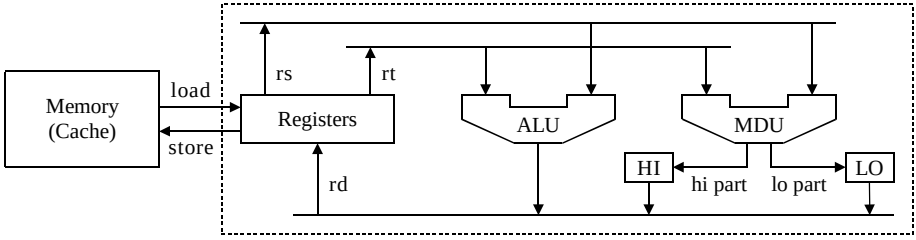

Fig. 2. Integer Unit (ALU) and Multiply/Divide Unit (MDU) of the 4Km.

MIPS processors implement a *delay slot* for load instructions, i.e. loads require extra cycles to complete before they exit the pipeline. For this reason, the instruction after the load must not "use" the result of the load instruction. MIPS branch instructions' effects are also delayed by one instruction; the instruction following the branch instruction is always executed, regardless of whether the branch is taken or not. The "bare" MIPS32 processors support a single addressing mode: *Indexed addressing*. In indexed addressing, an offset is encoded in the instruction word along with a base register. The offset is added to the base register's contents to form an effective address. Indexed addressing is useful when incrementing through the elements of an array as the addresses can be easily constructed at run-time.

3.1 MDU Pipeline of the 4Km Core

The 4Km processor core contains an autonomous multiply/divide unit (MDU) with a separate pipeline for multiply, multiply-and-add, and divide operations (see Figure 2). This pipeline operates in parallel with the integer unit (IU) pipeline and does not necessarily stall when the IU pipeline stalls (and vice versa). Long-running (multi-cycle) MDU operations, such as a divide, can be partially masked by other integer unit instructions.

The 4Km MDU consists of a (32×16)-bit Booth recoded[3] multiplier, two result/accumulation registers (referenced by the names `HI` and `LO`), a divide state machine, and the necessary control logic. The MIPS32 architecture defines the result of a multiply operation to be placed in the `HI` and `LO` registers. Using `MFHI` (move from HI) and `MFLO` (move from LO) instructions, these values can be transferred to general-purpose registers. The MIPS32 has also a "multiply-and-add" (`MADD`) instruction, which multiplies two 32-bit words and adds the product to the 64-bit concatenated values in the `HI/LO` register pair. Then, the resulting value is written back to the `HI` and `LO` registers. Various signal processing routines can make heavy use of that instruction, which optimizing compilers automatically generate when appropriate.

[3] Modified Booth recoding halves the number of partial products by using a signed digit radix-4 representation for one of the operands (see [4] for more details).

The targeted multiply instruction `MUL` (multiply with register write) directs its result to a GPR instead of the `HI/LO` register pair. All other multiply and multiply-and-add operations write to the `HI/LO` register pair. The integer operations, on the other hand, write to the general-purpose registers. Because MDU operations write to different registers than integer operations, following integer instructions can execute before the MDU operation has completed [21].

In our previous work [11] we demonstrated that an Assembly implementation of the FIOS inner loop requires (at least) 20 "native" MIPS32 instructions. Consequently, the FIOS inner loop can not be executed in less than 20 clock cycles, even if we assume that the 4Km is a "perfect" RISC processor without pipeline stalls, load or branch delays, and cache misses. A straightforward Assembly implementation of the FIPS inner loop would require even more cycles since the 64-bit accumulator of the 4Km is rather unappropriate for multiple-precision arithmetic. However, these shortcomings can be remedied by simple enhancements of the processor core, which will be demonstrated in the following sections. More precisely, augmenting the 4Km core with a "wide" accumulator allows to execute the FIPS inner loop in nine clock cycles.

4 Architectural Support for the FIPS Method

The FIPS Montgomery multiplication (Algorithm 1) comprises two nested loops with identical inner loop operations. Each iteration of the inner loop performs two multiplications and adds the products to a cumulative sum. Therefore, the FIPS method is very efficient on processors with a multiply/accumulate (MAC) unit. In this section we propose a simple enhancement of the 4Km MAC unit and two custom instructions for the FIPS method.

The MAC unit of the 4Km was designed with having DSP/multimedia workloads in mind. Signal processing routines mostly perform operations on small integers (e.g. 8-bit pixel color values, 16-bit audio samples), which means that a 64-bit accumulator serves its purpose perfectly well. However, in long integer arithmetic we want to exploit the full 32-bit precision of the registers. Comba's method and the FIPS Montgomery multiplication would profit from a "wide" accumulator so that a certain number of 64-bit products can be summed up without loss of precision. For instance, extending the accumulator by eight guard bits means that we can accumulate up to 256 products, which is sufficient for a 2048-bit Montgomery multiplication when $w = 32$. Moreover, register `HI` must be able to accommodate 40 bits instead of 32. The extra hardware cost is negligible, and a slightly longer critical path in the MAC's final adder is irrelevant for most applications, especially for smart cards.

The `MADDU` (*Multiply and Add Unsigned*) instruction multiplies two 32-bit words, treating them as unsigned integers, and adds the product to the concatenated values in the `HI/LO` register pair. Then, the resulting value is written back to the `HI` and `LO` registers [22]. This is exactly the operation performed twice in the inner loop of the FIPS Montgomery multiplication. Although `MADDU` is a native MIPS32 instruction, we nonetheless need two custom instructions for the

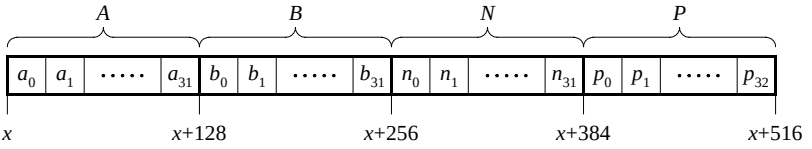

Fig. 3. Memory location of multiple-precision integers A, B, N, and P

FIPS method to perform well. The first instruction is used in the outer loop, and the second instruction facilitates the FIPS Montgomery squaring.

4.1 First Proposed Instruction: SHA

The `MFHI` (*Move from HI*) instruction copies only the least significant 32 bits of the `HI` register to the destination register. Of course, this raises the question of how to access the guard bits. A simple solution is to augment the processor with a custom instruction for shifting the concatenated values in the `HI/LO` register pair 32 bits to the right (with zeroes shifted in). We call this instruction `SHA`, which stands for *Shift Accumulator value*. The `SHA` instruction can be used to perform the operations at lines 10 and 18 of Algorithm 1. Executing `SHA` copies the contents of `HI` to `LO` and the eight guard bits to `HI`. The hardware cost of the `SHA` instruction is negligible.

4.2 Second Proposed Instruction: M2ADDU

The product scanning technique is not only applicable to Montgomery multiplication but also to Montgomery squaring. However, the "finely" integration of squaring and reduction is most effective when the processor offers an instruction for calculations of the form $2 \times a \times b + S$. This instruction, which we call `M2ADDU`, multiplies two 32-bit quantities, doubles the product, and accumulates it to the concatenated values in the `HI/LO` register pair. The multiplication by 2 is simply realized via a hard-wired left shift and requires essentially no additional hardware (except of a few multiplexors).

The availability of `M2ADDU` makes the optimization of squaring easier. FIPS Montgomery squaring is approximately 18% faster than FIPS Montgomery multiplication. This is because reduction requires always the same effort, regardless of whether it is integrated into multiplication or squaring.

4.3 Optimized Assembly Code for the FIPS Inner Loop

Long integer arithmetic is generally characterized by a large number of memory accesses since operands of very high precision (e.g. 1024 bits) can not be kept in the register file. Therefore, it is desirable to minimize the overhead caused by address calculations. In our implementation, we place the multiple-precision operands A, B, N, and P in contiguous locations in memory, as illustrated in Figure 3 for $n = 1024$ bits (each word consists of four bytes).

```
Loop:  LW     Rb, 0(Rk)     # load B[i-j] into Rb
       LW     Ra, 0(Rj)     # load A[j] into Ra
       LW     Rn, 128(Rk)   # load N[i-j] into Rn
       MADDU  Ra, Rb        # (HI,LO) += Ra * Rb
       LW     Rp, 384(Rj)   # load P[j] into Rp
       ADDIU  Rk, Rk, -4    # Rk = Rk - 4
       MADDU  Rp, Rn        # (HI,LO) += Rp * Rn
       BNE    Rk, Rz, Loop  # branch if Rk != Rz
       ADDIU  Rj, Rj, 4     # Rj = Rj + 4
```

Fig. 4. Inner loop of the FIPS Montgomery multiplication

Figure 4 shows an Assembly routine for the inner loop of the FIPS Montgomery multiplication (Algorithm 1). The code is optimized for an operand length of $n = 1024$ bits and takes advantage of the indexed addressing mode for fast address calculation. Our implementation starts with `LW` instructions to load the operands a_j and b_{i-j} into two general-purpose registers. The first `MADDU` instruction computes the product $a_j \cdot b_{i-j}$ and accumulates it to a running total stored in the `HI/LO` register pair. Note that the extended precision of the accumulator and the `HI` register guarantee that there is no overflow or loss of precision. The operands p_j and n_{i-j} are loaded immediately before and after the first `MADDU` instruction. Two `ADDIU` instructions, which implement simple pointer arithmetic, are used to fill a load and branch delay slot, respectively. The second `MADDU` is executed immediately before the branch instruction. There is no `SW` (store word) instruction in the inner loop since the memory write operations take place at the outer loop. The instruction sequence depicted in Figure 4 is carefully ordered to avoid pipeline stalls caused by load or branch delays.

Algorithm 1 moves through the individual words of A and P in ascending order and through the words of B and N in descending order. Therefore, we maintain two pointers in the inner loop; the first one is stored in register `Rj` and points to the current word a_j, and the second one is stored in `Rk` and points to b_{i-j}. The contents of register `Rj` is incremented by 4 each time the loop repeats, whereas `Rk` is decremented by 4. For $n = 1024$, the offset between a_j and p_j is exactly 384 bytes, and the offset between b_{i-j} and n_{i-j} is 128 bytes (see Figure 3). The current addresses of p_j and n_{i-j} can be easily constructed at run-time with help of the indexed addressing mode. Note that the contents of register `Rk` is also used to test the loop termination condition. Before entering the inner loop, register `Rz` is initialized with the address of a_{31}. Thus, the inner loop is iterated exactly $i - 1$ times[4].

Algorithm 1 contains two inner loops. The second j-loop is almost identical to the first one described before, except that the registers `Rj`, `Rk`, and `Rz` have to be initialized with different addresses. Moreover, the branch instruction must be adapted accordingly, i.e. register `Rj` has to be used to determine the loop

[4] It is not necessary to maintain an extra loop count variable since we can also use the address pointers for that purpose.

Table 1. Comparison of architectural enhancements for public-key cryptography

Implementation	1024-bit Mod. mul. # cycles	1024-bit Mod. squ. # cycles	1024-bit Mod. exp. ms (MHz)	Exp. alg.	CRT
Dhem [6]	??	??	480 (32)	Slid. wd.	Yes
Phillips *et al.* [27]	??	??	875 (25)	Slid. wd.	Yes
Prev. work [11]	11,500	9,700	800 (20)	Binary	No
This work	10,300	8,500	425 (33)	Binary	No

termination. Our custom instruction `SHA` is applicable in the outer loop at lines 10 and 18, respectively. The operation at line 8 of Algorithm 1 can be performed with the targeted multiply instruction, `MUL`, which calculates only the lower part of a product and writes it to a general-purpose register[5].

For a simple estimation of the execution time let us assume that the processor is equipped with a fully-parallel (32×32)-bit multiplier able to execute the `MADDU` instruction in a single clock cycle. In this case, the inner loop of the FIPS Montgomery multiplication requires nine cycles for one iteration if there are no cache misses. However, a major advantage of the FIPS method is that it does not need a single-cycle multiplier to reach peak performance (which is not the case with FIOS — see [11]). The influence of the multiplier latency will be discussed in Section 5.2. An optimized implementation of FIPS Montgomery squaring is almost 18% faster than generic FIPS Montgomery multiplication.

5 Simulation Results and Discussion

We used the SystemC language [29] to develop a functional model of the extended MIPS32 architecture and the software algorithms. First, the algorithms were coded in plain C with the inner loop operations modelled at a high abstraction level. Then, the inner loops were refined to Assembly instructions and their execution was simulated on a cycle-accurate model of the extended MIPS32 core.

5.1 Performance and Hardware Cost

Given a single-cycle multiplier, our simulations demonstrate that a 1024-bit FIPS Montgomery multiplication can be executed in 10,300 clock cycles. The custom instruction `M2ADDU` makes FIPS Montgomery squaring almost 18% faster, i.e. 8,500 cycles. Thus, a 1024-bit modular exponentiation according to the binary method can be performed in about $14 \cdot 10^6$ clock cycles, which corresponds to an execution time of 425 msec when the processor is clocked at 33 MHz. A comparison with related work (see Table 1) clearly demonstrates the efficiency

[5] Reference [22] says that the contents of `HI` and `LO` are unpredictable after a `MUL` instruction. However, Algorithm 1 requires that `MUL` does not overwrite the registers `HI` and `LO`. This has to be considered when using multi-cycle multiplier.

Table 2. Performance of 32-bit RISC cores with crypto extensions (at 33 MHz)

Company	Product	1024-bit RSA
ARM Limited	SecurCore SC200 [2]	594 msec
MIPS Technologies, Inc.	SmartMIPS 4KSc [23,19]	320 msec
NEC Electronics, Inc.	V-WAY32 μPD79215000 [26]	436 msec
STMicroelectronics	SmartJ ST22XJ64 [28]	380 msec

of our proposed extensions, the more so as the timings reported in [6,27] were achieved with a sliding window technique for exponentiation. In our previous work [11], we proposed two custom instructions, `MADDH` and `MADDL`, to accelerate Montgomery multiplication according to the FIOS method. These instructions allow to execute the FIOS inner loop in ten clock cycles, which results in the timings shown in Table 1. However, the performance of the FIOS method depends heavily on the multiplier latency (see next subsection). We point out that an execution time of 425 msec for a 1024-bit modular exponentiation is comparable to the performance of the commercial products specified in Table 2.

Architectural upgrades required to implement instruction set extensions are mainly localized to the Instruction Decode (ID) and Execute (EXE) pipeline stages [12]. The approach presented in this paper entails the addition of only two custom instructions; therefore the extra control logic for the instruction decoder is marginal. A conventional $(32 \times 32 + 64)$-bit multiply/accumulate unit can be easily equipped with a wide accumulator — a slight increase in area and delay is tolerable for most applications. The transistor count of a fully-parallel (single-cycle) (32×32)-bit Booth-recoded multiplier is roughly 28,000 [4].

Montgomery multiply and square operations produce different power traces. A careless implementation of the modular exponentiation could be exploited by an attacker to distinguish squares from multiplies. That way, the attacker can obtain the bits of the private key if the modular exponentiation is performed according to the binary exponentiation method. Some countermeasures against side-channel analysis have been proposed in the recent past, see e.g. [16,30].

5.2 Influence of the Multiplier Latency

The inner loop of the FIPS method depicted in Figure 4 executes two `MADDU` instructions. Note that the result of a `MADDU` operation is written to the `HI/LO` register pair, i.e. `MADDU` operations do not occupy the write port of the register file (see Figure 2). Therefore, `MADDU` has an "empty" slot when it is executed on a (32×16)-bit multiplier since neither the register file's read ports nor the write port are occupied during the second cycle. This means that other arithmetic/logical instructions can be executed during the latency period of the `MADDU` operation. In other words, `MADDU` does not stall the IU pipeline (even when it is realized as a multi-cycle instruction) as long as there are independent instructions available which do not use the result of `MADDU`.

The FIPS method allows to mask the latency period of the `MADDU` operations if we order the instruction sequence properly. Our experimental results indicate that a (32×16)-bit multiplier also allows to execute the inner loop of the FIPS method in nine clock cycles. Even a MIPS32 core with a (32×12)-bit multiplier[6] would not require more than nine cycles to execute the Assembly code shown in Figure 4. This clearly demonstrates that the FIPS method does not necessarily require a single-cycle multiplier to reach peak performance. For instance, a fully parallel (32×32)-bit multiplier is not able to execute the FIPS inner loop faster than a serial/parallel multiplier which requires three clock cycles to complete a `MADDU` operation.

On the other hand, the performance of the architectural enhancements for the FIOS method proposed in [11] depends heavily on the latency of the multiplier. The two custom instructions `MADDH` and `MADDL` introduced in [11] perform multiply-and-add operations of the form $a \times b + c + d$ and write either the higher or the lower part of the result to a general-purpose register. If, for instance, the MIPS32 core contains a (32×16)-bit multiplier, then the (32×32)-bit operations take two clock cycles to complete. Both `MADDH` and `MADDL` occupy the read ports of register file during the first cycle and the write port during the second cycle. Consequently, no other arithmetic/logical instruction can be scheduled in parallel, which means that `MADDH` and `MADDL` always force a stall of the IU pipeline in order to maintain their register file write slot (during the write-back phase). A (32×16)-bit multiplier would increase the execution time of the FIOS inner loop by two clock cycles, i.e. any iteration of the inner loop takes twelve cycles instead of ten [11]. Therefore, a multi-cycle multiplier deteriorates the performance of the architectural enhancements for the FIOS method, which is not necessarily the case for the FIPS method.

5.3 Performance Scalability

An execution time of about 425 msec for a 1024-bit modular exponentiation is reasonable for smart card applications and tolerated by most users. The trivial way to further increase the performance is to crank up the clock speed. On the other hand, we can also increase the performance by dedicating more hardware resources. Processor customization offers many options to achieve this:

- Implement auto-increment/decrement addressing modes. This eliminates the need to perform the pointer arithmetic "by hand", making it possible to execute the FIPS inner loop in seven clock cycles, and a 1024-bit modular exponentiation in less than 350 msec.
- Perform memory accesses in parallel with other operations (execute one operation and simultaneously load operands for the next one from memory).
- Use a multiple bus architecture that allows multiple (parallel) memory accesses in one clock cycle. Several DSPs, e.g. Motorola 56k or Analog Devices 210x, have two memory banks which are accessible in parallel.

[6] Note that Intel's StrongARM SA-110 processor contains a (32×12)-bit multiplier.

- Hardware support for looping ("zero overhead looping") allows tight loops to be repeated without wasting time for updating and testing the loop counter.
- Use a number of MAC units in parallel, following an SIMD approach. The FIPS method allows to utilize an arbitrary number of execution units, without the carry propagation becoming a performance bottleneck.

There are also a variety of algorithmic and software-related optimizations to increase the performance. Examples for the former include the Chinese Remainder Theorem (for RSA private key operations) and the application of an advanced exponentiation method. Software optimization techniques may involve (partial) loop unrolling and the full use of available registers.

6 Summary of Results and Conclusions

In this paper, we introduced simple architectural enhancements to increase the software performance of Montgomery multiplication according to the FIPS method. We proposed two custom instructions and analyzed their performance on an extended MIPS32 core. Our experiments show that a 1024-bit modular exponentiation can be performed in 425 msec when the processor is clocked at 33 MHz. All presented concepts (i.e. the extended MIPS32 core and the software routines) have been verified by co-simulation with the SystemC language.

The proposed extensions blur the traditional line between general-purpose hardware (processor core) and application-specific hardware, thereby enabling fast yet flexible implementations of Montgomery multiplication. Moreover, the architectural enhancements entail only minor tweaks in the processor core and require almost no additional hardware (in relation to the hardware cost of a cryptographic co-processor). The extended core remains fully compatible to the MIPS32 architecture. Another benefit of the presented approach is that writing and debugging software is much cheaper than designing, implementing, and testing a cryptographic co-processor.

The execution time of a 1024-bit modular exponentiation can be reduced to 350 msec when the processors supports an auto-increment/decrement addressing mode. This result is comparable to the performance of commercial smart card RISC cores like the *SmartMIPS*, which requires, according to [19], approximately 320 msec at the same clock frequency. The major advantage of the proposed architectural enhancements is the fact that a single-cycle (32×32)-bit multiplier is not necessary to reach peak performance. Therefore, the FIPS method together with the wide accumulator approach allows to achieve a good trade-off between area and performance.

Acknowledgements. The work described in this paper origins from the European Commission funded project *Crypto Module with USB Interface (USB_CRYPT)* established under contract IST-2000-25169 in the Information Society Technologies (IST) Program.

References

1. ARC International. Technical summary of the ARCtangent™-A4 processor core. Product brief, available for download at `http://www.arc.com/upload/download/ARCIntl_0311_TechSummary_DS.pdf`, 2001.
2. ARM Limited. ARM SecurCore Solutions. Product brief, available for download at `http://www.arm.com/aboutarm/4XAFLB/$File/SecurCores.pdf`, 2002.
3. P. D. Barrett. Implementing the Rivest, Shamir and Adleman public-key encryption algorithm on a standard digital signal processor. In *Advances in Cryptology — CRYPTO '86*, vol. 263 of *Lecture Notes in Computer Science*, pp. 311–323. Springer Verlag, 1987.
4. K. Choi and M. Song. Design of a high performance 32×32-bit multiplier with a novel sign select Booth encoder. In *Proceedings of the 34th IEEE International Symposium on Circuits and Systems (ISCAS 2001)*, vol. II, pp. 701–704. IEEE, 2001.
5. P. G. Comba. Exponentiation cryptosystems on the IBM PC. *IBM Systems Journal*, 29(4):526–538, Oct. 1990.
6. J.-F. Dhem. *Design of an efficient public-key cryptographic library for RISC-based smart cards*. Ph.D. Thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 1998.
7. S. R. Dussé and B. S. Kaliski. A cryptographic library for the Motorola DSP56000. In *Advances in Cryptology — EUROCRYPT '90*, vol. 473 of *Lecture Notes in Computer Science*, pp. 230–244. Springer Verlag, 1991.
8. P. Faraboschi, G. M. Brown, J. A. Fisher, G. Desoli, and M. O. Homewood. Lx: A technology platform for customizable VLIW embedded processing. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA 2000)*, pp. 203–213. ACM Press, 2000.
9. R. E. Gonzalez. Xtensa: A configurable and extensible processor. *IEEE Micro*, 20(2):60–70, Mar./Apr. 2000.
10. J. R. Goodman. *Energy Scalable Reconfigurable Cryptographic Hardware for Portable Applications*. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.
11. J. Großschädl and G.-A. Kamendje. Optimized RISC architecture for multiple-precision modular arithmetic. In *Security in Pervasive Computing — SPC 2003*, vol. 2802 of *Lecture Notes in Computer Science*. Springer Verlag, 2003 (in print).
12. M. Gschwind. Instruction set selection for ASIP design. In *Proceedings of the 7th International Symposium on Hardware/Software Codesign (CODES '99)*, pp. 7–11. ACM Press, 1999.
13. K. W. Keutzer, S. Malik, and A. R. Newton. From ASIC to ASIP: The next design discontinuity. In *Proceedings of the 20th International Conference on Computer Design (ICCD 2002)*, pp. 84–90. IEEE Computer Society Press, 2002.
14. D. E. Knuth. *Seminumerical Algorithms*, vol. 2 of *The Art of Computer Programming*. Addison-Wesley, 3rd edition, 1998.

15. Ç. K. Koç, T. Acar, and B. S. Kaliski. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3):26–33, June 1996.
16. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology — CRYPTO '96*, vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113. Springer Verlag, 1996.
17. K. Küçükçakar. An ASIP design methodology for embedded systems. In *Proceedings of the 7th International Symposium on Hardware/Software Codesign (CODES '99)*, pp. 17–21. ACM Press, 1999.
18. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
19. MIPS Technologies, Inc. Making smart cards secure. *The Pipeline (Technology Newsletter)*, Fall 2001, p. 4. Available for download at `http://www.mips.com/content/PressRoom/Newsletter`, 2001.
20. MIPS Technologies, Inc. MIPS32 4Km™ processor core family data sheet. Available for download at `http://www.mips.com/publications/index.html`, 2001.
21. MIPS Technologies, Inc. MIPS32 4K™ processor core family software user's manual. Available for download at `http://www.mips.com/publications/index.html`, 2001.
22. MIPS Technologies, Inc. MIPS32™ architecture for programmers, Vol. I & II. Available for download at `http://www.mips.com/publications/index.html`, 2001.
23. MIPS Technologies, Inc. SmartMIPS Architecture Smart Card Extensions. Product brief, available for download at `http://www.mips.com/ProductCatalog/P_SmartMIPSASE/SmartMIPS.pdf`, 2001.
24. MIPS Technologies, Inc. Pro Series™ Processor Cores. Product brief, available for download at `http://www.mips.com/ProductCatalog/P_ProSeriesFamily/proseries.pdf`, 2003.
25. P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, Apr. 1985.
26. NEC Electronics, Inc. V-WAY32 32-bit Security Cryptocontroller. Product letter, available for download at `http://www.nec.com.sg/es/Smartcard.htm`, 2000.
27. B. J. Phillips and N. Burgess. Implementing 1,024-bit RSA exponentiation on a 32-bit processor core. In *Proceedings of the 12th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2000)*, pp. 127–137. IEEE Computer Society Press, 2000.
28. STMicroelectronics. ST22 SmartJ Platform Smartcard ICs. Product brief, available for download at `http://www.st.com/stonline/prodpres/smarcard/insc9901.htm`, 2002.
29. The Open SystemC Initiative (OSCI). *SystemC Version 2.0 User's Guide*. Available for download at `http://www.systemc.org`, 2002.
30. C. D. Walter. MIST: An efficient, randomized exponentiation algorithm for resisting power analysis. In *Topics in Cryptology — CT-RSA 2002*, vol. 2271 of *Lecture Notes in Computer Science*, pp. 53–66. Springer Verlag, 2002.
31. A. Wang, E. Killian, D. E. Maydan, and C. Rowen. Hardware/software instruction set configurability for system-on-chip processors. In *Proceedings of the 38th Design Automation Conference (DAC 2001)*, pp. 184–188. ACM Press, 2001.

Author Index