

Sihan Qing  
Wenbo Mao  
Javier Lopez  
Guilin Wang (Eds.)

LNCS 3783

# Information and Communications Security

**7th International Conference, ICICS 2005  
Beijing, China, December 2005  
Proceedings**

 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*New York University, NY, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Moshe Y. Vardi

*Rice University, Houston, TX, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Sihan Qing Wenbo Mao Javier Lopez  
Guilin Wang (Eds.)

# Information and Communications Security

7th International Conference, ICICS 2005  
Beijing, China, December 10-13, 2005  
Proceedings



Springer

Volume Editors

Sihan Qing  
Chinese Academy of Sciences, Institute of Software  
Beijing 100080, P.R. China  
E-mail: qsihan@ercist.iscas.ac.cn

Wenbo Mao  
HP Labs. China  
112 Jian Guo Road, Beijing 100022, P.R. China

Javier Lopez  
University of Malaga, Computer Science Department  
29071 Malaga, Spain  
E-mail: jlm@lcc.uma.es

Guilin Wang  
Institute for Infocomm Research  
21 Heng Mui Keng Terrace, Singapore 119613  
E-mail: glwang@i2r.a-star.edu.sg

Library of Congress Control Number: 2005937067

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

ISSN 0302-9743  
ISBN-10 3-540-30934-9 Springer Berlin Heidelberg New York  
ISBN-13 978-3-540-30934-5 Springer Berlin Heidelberg New York

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

Springer is a part of Springer Science+Business Media  
springeronline.com

© Springer-Verlag Berlin Heidelberg 2005  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 11602897 06/3142 5 4 3 2 1 0

# Preface

The Seventh International Conference on Information and Communications Security, ICICS 2005, was held in Beijing, China, 10-13 December 2005. The ICICS conference series is an established forum for exchanging new research ideas and development results in the areas of information security and applied cryptography. The first event began here in Beijing in 1997. Since then the conference series has been interleaving its venues in China and the rest of the world: ICICS 1997 in Beijing, China; ICICS 1999 in Sydney, Australia; ICICS 2001 in Xi'an, China; ICICS 2002 in Singapore; ICICS 2003 in Hohhot City, China; and ICICS 2004 in Malaga, Spain. The conference proceedings of the past events have always been published by Springer in the Lecture Notes in Computer Science series, with volume numbers, respectively: LNCS 1334, LNCS 1726, LNCS 2229, LNCS 2513, LNCS 2836, and LNCS 3269.

ICICS 2005 was sponsored by the Chinese Academy of Sciences (CAS); the Beijing Natural Science Foundation of China under Grant No. 4052016; the National Natural Science Foundation of China under Grants No. 60083007 and No. 60573042; the National Grand Fundamental Research 973 Program of China under Grant No. G1999035802, and Hewlett-Packard Laboratories, China. The conference was organized and hosted by the Engineering Research Center for Information Security Technology of the Chinese Academy of Sciences (ERCIST, CAS) in co-operation with the International Communications and Information Security Association (ICISA).

The aim of the ICICS conference series has been to offer the attendees the opportunity to discuss the latest developments in theoretical and practical aspects of information and communications security. The Technical Program for this year had three parts: (1) paper presentations, which consisted of 40 papers selected from 235 submissions, (2) two invited speeches, one from academia by Prof. Jean-Jacques Quisquater of the University of Louvain and one from industry by Mr. Graeme Proudler of Hewlett-Packard Laboratories, Bristol and Chairman of the Technical Committee of the Trusted Computing Group, and (3) Trusted Computing Technical Presentations (TCTP@ICICS 2005), which consisted of Trusted Computing solutions and demo showcases presented by Trusted Computing technology providers from industry. TC, which is defined, specified and promoted by the industry standard body Trusted Computing Group (TCG), is an important and pervasively progressing topic in platform security. However, it has so far mainly been researched and developed in industry. We believe that a closer involvement in TC from academia will help to advance this important area. TCTP@ICICS 2005 aimed to enhance interactions between academia and industry on the topic of TC.

We are grateful to the program committee members and external referees for their precious time and valued contribution to the tough and time-consuming

review process. We are also pleased to thank Dr. Guilin Wang for his great help in publishing affairs, Dr. Jianbo He for his great contribution to website related affairs, and Mr. Yinghe Jia, Prof. Yeping He, Prof. Xizhen Ni, and other members of the Organizing Committee for helping with many local details.

Finally we wish to thank the authors of every paper, whether accepted or not, the attendees of the conference and all the other people who contributed to the conference in various ways.

September 2005

Sihan Qing  
Wenbo Mao  
Javier Lopez

# ICICS 2005

## Seventh International Conference on Information and Communications Security

Beijing, China  
December 10-13, 2005

*Organized by*

Engineering Research Center for Information Security Technology (ERCIST)  
(Chinese Academy of Sciences)

*In co-operation with*

International Communications and Information Security Association (ICISA)

*Sponsored by*

Chinese Academy of Sciences (CAS)  
Beijing Natural Science Foundation of China  
National Natural Science Foundation of China  
National Grand Fundamental Research 973 Program of China *and*  
Hewlett-Packard Laboratories, China

### General Chair

Sihan Qing Chinese Academy of Sciences, China

### Program Chairs

Sihan Qing Chinese Academy of Sciences, China  
Wenbo Mao HP Labs, Beijing & Bristol  
Javier Lopez University of Malaga, Spain

### Program Committee

Tuomas Aura Microsoft, UK  
Feng Bao Institute for Infocomm Research, Singapore  
Alex Biryukov Katholieke Univ. Leuven, Belgium  
Mike Burmester Florida State University, USA  
Chin-Chen Chang National Chung Cheng University, Taiwan  
Lily Chen Motorola, USA  
Welland Chu Thales, Hong Kong, China

## VIII Organization

Bruno Crispo	Vrije University, Holland
Ed Dawson	Queensland University of Technology, Australia
Robert H. Deng	Singapore Management University, Singapore
Yvo Desmedt	University College London, UK
Josep Domingo-Ferrer	Univ. Rovira-Virgili, Spain
Dengguo Feng	Chinese Academy of Sciences, China
Antonio Gomez-Skarmeta	Univ. of Murcia, Spain
Stefanos Gritzalis	University of Aegean, Greece
Yongfei Han	Onets, China
Hai Jin	Huazhong Univ. of Sci. & Tech., China
Marc Joye	Gemplus & CIM-PACA, France
Kwangjo Kim	Information and Communications University, Korea
Chi-Sung Laih	National Cheng Kung University, Taiwan
Antonio Maña	University of Malaga, Spain
Catherine Meadows	Naval Research Laboratory, USA
Eiji Okamoto	University of Tsukuba, Japan
Giuseppe Persiano	Università di Salerno, Italy
David Pointcheval	ENS, France
Jean-Jacques Quisquater	UCL, Belgium
Bimal Roy	Indian Statistical Institute, India
Rei Safavi-Naini	University of Wollongong, Australia
Kouichi Sakurai	Kyushu University, Japan
Tomas Sander	HP Labs, Princeton, USA
Nigel Smart	Bristol University, UK
Miguel Soriano	UPC, Spain
Vijay Varadharajan	Macquarie University, Australia
Guozhen Xiao	Xidian University, China
Yiqun Lisa Yin	Independent security consultant, USA
Moti Yung	Columbia University & RSA Labs, USA
Yuliang Zheng	University of North Carolina at Charlotte, USA
Jianying Zhou	Institute for Infocomm Research, Singapore

### Publication Chair

Guilin Wang	Institute for Infocomm Research, Singapore
-------------	--

### Organizing Committee Chairs

Yinghe Jia	China Information Security Technology Committee, China
Yeping He	ERCIST, Chinese Academy of Sciences, China
Xizhen Ni	ERCIST, Chinese Academy of Sciences, China



**External Reviewers**

Joonsang Baek	Venkat Balakrishnan	T. Balopoulos
Paulo Barreto	Kemal Bicakci	Colin Boyd
Xavier Boyen	Hongxu Cai	Oscar Canovas
Alvaro Cardenas	Jordi Castellà-Roca	Dario Catalano
Julien Cathalo	Debrup Chakraborty	Sanjit Chatterjee
Yongxi Cheng	Zhian Cheng	Andrew Clark
Félix J. García Clemente	Scott Contini	Paolo D'Arco
Xuhua Ding	Yevgeniy Dodis	Qingkuan Dong
Boris Dragovic	Jiang Du	Dang Nguyen Duc
Ratna Dutta	Oscar Esparza	Marcel Fernandez
Jordi Forne	Xiaotong Fu	Clemente Galdi
Chandana Gamage	Jie Guo	Lifeng Guo
L. Gymnopoulos	Shai Halevi	Yong-Sork Her
Juan Hernández-Serrano	Yoshiaki Hori	John Iliadis
Kenji Imamoto	Sarath Indrakanti	C. Kalloniatis
Georgios Kambourakis	HyunChan Kim	Costas Lambrinoudakis
Tri V. Le	Eonkyung Lee	Hyunrok Lee
Dimitrios Lekkas	Manuel Leone	Jung-Shian Li
Minming Li	Ninghui Li	Shengqiang Li
Benoît Libert	Vo Duc Liem	Chi-Jen Lu
Wenming Lu	Miao Ma	Antoni Martínez-Ballesté
Barbara Masucci	Gabriel López Millán	José L. Muñoz-Tapia
Aarthi Nagarajan	Gregory Neven	Svetla Nikova
Peng Ning	Ryuzo Nishi	Elisabeth Oswald
Dan Page	Pascal Paillier	Jae Min Park
Josep Pegueroles	Kun Peng	Olivier Pereira
Gregorio Martinez Perez	Angela Piper	Bogdan Popescu
Chun Ruan	Palash Sarkar	Jasper Scholten
Francesc Sebé	Wook Shin	Agusti Solanas
Martijn Stam	François-Xavier Standaert	Gelareh Taban
Dongyu Tonien	Udaya Kiran Tupakula	Yoshifumi Ueshige
Frederik Vercauteren	Ivan Visconti	Zhiguo Wan
Chen Wang	Shuhong Wang	Yongdong Wu
Jing Xiao	JonPhil Yang	Yanjiang Yang
Janson Zhang	Jing Zhang	Ning Zhang
Weiliang Zhao	Yingchao Zhao	Yunlei Zhao
Huafei Zhu		

# Table of Contents

## Fair Exchange

An Evenhanded Certified Email System for Contract Signing <i>Kenji Imamoto, Jianying Zhou, Kouichi Sakurai</i> .....	1
Efficient ID-Based Optimistic Fair Exchange with Provable Security <i>Zhenfeng Zhang, Dengguo Feng, Jing Xu, Yongbin Zhou</i> .....	14
On the Quest for Impartiality: Design and Analysis of a Fair Non-repudiation Protocol <i>J. Cederquist, R. Corin, M. Torabi Dashti</i> .....	27
Generic, Optimistic, and Efficient Schemes for Fair Certified Email Delivery <i>Guilin Wang, Feng Bao, Kenji Imamoto, Kouichi Sakurai</i> .....	40

## Digital Signatures I

Cryptanalysis of a Forward Secure Blind Signature Scheme with Provable Security <i>Shuhong Wang, Feng Bao, Robert H. Deng</i> .....	53
On Delegatability of Four Designated Verifier Signatures <i>Yong Li, Helger Lipmaa, Dingyi Pei</i> .....	61
PIATS: A Partially Sanitizable Signature Scheme <i>Tetsuya Izu, Nobuyuki Kanaya, Masahiko Takenaka, Takashi Yoshioka</i> .....	72

## Cryptographic Protocols

Ciphertext Comparison, a New Solution to the Millionaire Problem <i>Kun Peng, Colin Boyd, Ed Dawson, Byoungcheon Lee</i> .....	84
Private Itemset Support Counting <i>Sven Laur, Helger Lipmaa, Taneli Mielikäinen</i> .....	97
Visual Cryptographic Protocols Using the Trusted Initializer <i>Hidenori Kuwakado, Masakatu Morii, Hatsukazu Tanaka</i> .....	112

Admissible Interference by Typing for Cryptographic Protocols  
*Alaaeddine Fellah, John Mullins* ..... 123

**Cryptanalysis**

On the Security Bounds of CMC, EME, EME<sup>+</sup> and EME\* Modes of Operation  
*Raphael C.-W. Phan, Bok-Min Goi* ..... 136

On the Security of Encryption Modes of MD4, MD5 and HAVAL  
*Jongsung Kim, Alex Biryukov, Bart Preneel, Sangjin Lee* ..... 147

Cryptanalysis of PASS II and MiniPass  
*Bok-Min Goi, Jintai Ding, M.U. Siddiqi* ..... 159

Simple Power Analysis on Fast Modular Reduction with NIST Recommended Elliptic Curves  
*Yasuyuki Sakai, Kouichi Sakurai* ..... 169

**Digital Signatures II**

Asymmetric Concurrent Signatures  
*Khanh Nguyen* ..... 181

Generic Construction of (Identity-Based) Perfect Concurrent Signatures  
*Sherman S.M. Chow, Willy Susilo* ..... 194

Sequential Aggregate Signatures Working over Independent Homomorphic Trapdoor One-Way Permutation Domains  
*Huafei Zhu, Feng Bao, Robert H. Deng* ..... 207

**Network Security**

Session Table Architecture for Defending SYN Flood Attack  
*Xin Li, Zhenzhou Ji, Mingzeng Hu* ..... 220

A Behavior-Based Ingress Rate-Limiting Mechanism Against DoS/DDoS Attacks  
*Song Huang, Ling Zhang, Shou-Ling Dong* ..... 231

Port Scan Behavior Diagnosis by Clustering  
*Lanjia Wang, Haixin Duan, Xing Li* ..... 243

Network Vulnerability Analysis Through Vulnerability Take-Grant Model (VTG) <i>Hamid Reza Shahriari, Reza Sadoddin, Rasool Jalili, Reza Zakeri, Ali Reza Omidian</i> . . . . .	256
---	-----

## Applied Cryptography

Multiplex Encryption: A Practical Approach to Encrypting Multi-recipient Emails <i>Wei Wei, Xuhua Ding, Kefei Chen</i> . . . . .	269
Secure Multicast Using Proxy Encryption <i>Yun-Peng Chiu, Chin-Laung Lei, Chun-Ying Huang</i> . . . . .	280
Efficient and Non-interactive Timed-Release Encryption <i>Julien Cathalo, Benoît Libert, Jean-Jacques Quisquater</i> . . . . .	291

## Key Management

Security Properties of Two Authenticated Conference Key Agreement Protocols <i>Qiang Tang, Chris J. Mitchell</i> . . . . .	304
Cryptanalysis of Two User Identification Schemes with Key Distribution Preserving Anonymity <i>Eun-Jun Yoon, Kee-Young Yoo</i> . . . . .	315
Enhanced ID-Based Authenticated Key Agreement Protocols for a Multiple Independent PKG Environment <i>Sangjin Kim, Hoonjung Lee, Heekuck Oh</i> . . . . .	323

## Access Control

Enforce Mandatory Access Control Policy on XML Documents <i>Lan Li, Xinghao Jiang, Jianhua Li</i> . . . . .	336
Network Access Control for Mobile Ad-Hoc Networks <i>Pan Wang, Peng Ning, Douglas S. Reeves</i> . . . . .	350
Remotely Keyed Cryptographics Secure Remote Display Access Using (Mostly) Untrusted Hardware <i>Debra L. Cook, Ricardo Baratto, Angelos D. Keromytis</i> . . . . .	363

## Applications

Authenticating Query Results in Data Publishing <i>Di Ma, Robert H. Deng, Hweehwa Pang, Jianying Zhou</i> . . . . .	376
Multi-Source Stream Authentication Framework in Case of Composite MPEG-4 Stream <i>Tieyan Li, Huafei Zhu, Yongdong Wu</i> . . . . .	389
Batching SSL/TLS Handshake Improved <i>Fang Qi, Weijia Jia, Feng Bao, Yongdong Wu</i> . . . . .	402
Achieving Efficient Conjunctive Keyword Searches over Encrypted Data <i>Lucas Ballard, Seny Kamara, Fabian Monrose</i> . . . . .	414

## Watermarking

Total Disclosure of the Embedding and Detection Algorithms for a Secure Digital Watermarking Scheme for Audio <i>David Megías, Jordi Herrera-Joancomartí, Julià Minguillón</i> . . . . .	427
Reversible Watermark with Large Capacity Using the Predictive Coding <i>Minoru Kuribayashi, Masakatu Morii, Hatsukazu Tanaka</i> . . . . .	441

## System Security

PCAV: Internet Attack Visualization on Parallel Coordinates <i>Hyunsang Choi, Heejo Lee</i> . . . . .	454
Implementation of Packet Filter Configurations Anomaly Detection System with SIERRA <i>Yi Yin, R.S. Bhuvaneswaran, Yoshiaki Katayama, Naohisa Takahashi</i> . . . . .	467
D_DIPS: An Intrusion Prevention System for Database Security <i>Jiazhu Dai, Huaikou Miao</i> . . . . .	481
<b>Author Index</b> . . . . .	491

# An Evenhanded Certified Email System for Contract Signing

Kenji Imamoto<sup>1</sup>, Jianying Zhou<sup>2</sup>, and Kouichi Sakurai<sup>1</sup>

<sup>1</sup> Information Science and Electrical Engineering, Kyushu University,  
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan  
imamoto@itslab.csce.kyushu-u.ac.jp, sakurai@csce.kyushu-u.ac.jp

<sup>2</sup> Institute for Infocomm Research,  
21 Heng Mui Keng Terrace, Singapore 119613  
jyzhou@i2r.a-star.edu.sg

**Abstract.** Certified email is a system which enables a sender to prove a receiver's receipt of email. Such a system can be used for applications related to electronic commerce on the Internet. This paper considers a situation where a sender or a receiver wants to change his/her mind due to the change of mail content value (e.g., stock, auction, gambling) during the transaction. We point out that no traditional certified email systems have been designed for such a case, thus one of the participants can be at a disadvantage. To avoid this problem, we propose an evenhanded certified email system in which each participant can change his/her choice, either cancel or finish the transaction, at any time during the transaction.

## 1 Introduction

As the Internet has become more and more popular, many contracts are being signed online as well. A variant of the contract signing problem is *certified email* in which, Alice sends a mail to Bob and wants some evidence (i.e., a receipt) that Bob received her mail. *Fairness* is an important requirement for a certified email protocol that guarantees when the protocol terminates, either both parties have obtained their desired items, or neither acquired any useful information.

Many certified email systems have been proposed [1, 2, 3, 4, 5, 6], and some systems are commercialized. For efficiency, most of certified email systems include a *trusted third party* (TTP) as a mediator. This mediator is involved to ensure the fairness of a transaction. Although protocols without TTP were also proposed [7], they are not practical in terms of computation and communication overheads. Hence, this paper only focuses on certified email systems that use a TTP.

Some of existing systems introduced the concept of *timeliness* (i.e., any participant can terminate a session in finite time without loss of fairness) to avoid waiting the other's response forever [1, 2]. In order to provide timeliness, a system proposed in [1] has two sub-protocols: *cancel protocol* and *help protocol*. By using the cancel protocol, a sender can cancel a session if the intended receiver ignores the sender's request. On the other hand, by using the help protocol, a

receiver can obtain the mail even if the sender does nothing after the receiver responded to the sender’s request.

In addition to termination of a session in finite time, the cancel protocol can also be used to change a sender’s decision before completing the session (i.e., she can stop transmission of the mail). However, since the receiver cannot execute the cancel protocol, he cannot change his mind after a particular point even before completing the session (i.e., he cannot deny the receipt of the mail). This difference leads to the following disadvantageous situation.

Suppose Alice sells gold to Bob at 100 dollars. The receipt of gold means that “Alice must sell gold at 100 dollars and Bob must pay 100 dollars”. Then, they execute the proposed certified email protocol to exchange gold and the receipt. After Bob’s response to Alice’s request (but before completing the protocol), someone who says “I want to buy gold at 120 dollars” might appear. In this case, she will cancel the protocol, and sell gold to the new buyer. On the other hand, someone who says “I want to sell gold at 80 dollars” might appear. In this case, Bob wants to cancel the protocol and buy gold from the new seller, but he cannot.

The above situation can happen frequently in the case of a contract that the item’s value is changeable, for example a soccer pool where the news of a player’s sudden injury may change the betters’ choices. Since no traditional certified email systems have been designed for such a case, one of the participants can be at a disadvantage. In this paper, we propose an *evenhanded system* in which each participant can change his/her choice anytime before a termination without loss of fairness. We call this property “*Change of Choice*”. In our proposed system, each participant has sub-processes to cope with any event in order to enjoy the best benefit. Note that each participant always plays in a way that increases his/her own benefit, and a participant who first acts will obtain the better benefit (i.e., first come, first served).

## 2 Model and Requirements

This paper considers contracts of changeable values, where a party owning a variable item wants to sell her item to another party. Suppose the digital item is delivered with a certified email system, and the sender can claim the payment (as indicated in the receipt) from the receiver by proving the item has been received by the receiver. To simplify our analysis, we assume the price offered at the starting point of a session is the one negotiated by both participants.

Each party communicates through a network where no message is lost or delayed, and the value of an exchanged item can change anytime. Each party decides his/her action to make own benefit as high as possible.

A standard certified email system has the following requirements.

- *Fairness*: Both participants can either obtain the result each one desires, or neither of them does.

- *Authentication*: Each participant can identify his/her partner.
- *Non-repudiation*: Both participants cannot repudiate their own action after the session is over.
- *Timeliness*: Either participant can terminate a session at any time without loss of fairness.

Non-repudiation has two variants. *Non-repudiation of receipt* guarantees that a receiver cannot deny the receipt of mail after the receiver actually received it. *Non-repudiation of origin* guarantees that a sender cannot deny the transmission of mail if the sender actually sent it.

Besides the above standard requirements, there is an extra requirement of “*Change of Choice*” for an evenhanded certified email system as described earlier.

### 3 Game Tree and Evenhanded System

To concretely define an evenhanded system that can avoid any one-sided disadvantageous scenario as described in Section 1, we define evenhanded situation and stage by using game theory in an extensive form<sup>1</sup>. In this section, we first introduce the notion of game theory, and next define the evenhanded system.

#### 3.1 System Expression Using Game Theory

Games, such as chess, can be represented by a labeled directed tree graph. Each vertex of this tree except for the leaves is labeled with the name of a player and represents a decision point for this player in the course of the game. The choices or possible moves at this point are represented by the edges starting from this vertex. The leaves of the tree correspond to possible ends of the game. Each leaf is labeled with a tuple of real numbers, which represent the payoffs for the players if the game ends in that leaf. The payoff may be negative, in which case it is interpreted as a loss. The starting point of the game is represented by the root of the tree. The goal of the players in a game is to maximize their payoffs.

There are some striking similarities between exchanges and games. Indeed, in both cases, we have two (or more) parties/players who interact with each other according to some rules, and whose actions influence the future actions of the other. From this reason, we try to express certified email systems by game tree using the following rules.

A system is advanced by one of the participants’ available choices. Generally, each participant may have one or more choices as follows: (1) execute faithfully, (2) make the session valid with the TTP’s help, (3) cancel the session, or (4) stay (i.e., do nothing). If both parties select the choice of stay, then the session is equivalent to being cancelled. A system consists of one or more “stages”, which are periods between each choice (except for stay) decided by one of the participants. Moreover, each stage has three “situations” divided by events related to

<sup>1</sup> A similar work in [8] defines fairness by game tree.



the changes of the exchanged item's value, that is, higher/lower than the initial value, or no change. No party can predict these events beforehand.

Each participant can execute anytime one of any available choices at each stage. This means it is not decided which party selects the choice first at each stage if both parties have choices. To express such a chance, we add a chance move (either a sender selects first or a receiver selects first) at the first node of each stage.

In the case of contract with changeable values, each participant's payoff depends on the change of value. For example, if the value becomes higher than the initial value, since the sender of the exchanged item suffers a loss by sending it at the initial value, the choice of cancelling the session can make a higher profit for the sender than another choice. On the other hand, receiving the item at the initial value makes a higher profit for the receiver. Contrary to the above situation, if the value becomes lower than the initial value, sending the item at the initial value makes a higher profit for the sender while cancelling the session makes a higher profit for the receiver. Therefore, we can say that payoff for a party is 1 if the end of a session is desired for the party, and payoff is  $-1$  if the end is not desired. However, completing a session by faithful execution can increase slightly both participants' payoffs ( $0 < p_f < 1$ )<sup>2</sup>.

### Meaning of Each Object in Game Tree

In the following figures, black circle denotes a chance move, white circle denotes a sender's turn, and gray circle denotes a receiver's turn. Each edge represents available choice (i.e.,  $F_i$  is  $i$ th procedure of the faithful execution,  $C$  is cancel,  $S$  is stay,  $H$  is TTP's help), and possible move (i.e.,  $S_F$  means that the sender can select a choice before the receiver, and  $R_F$  means the contrary) at each point. Also each square denotes a termination of the session, at which payoffs for both participants are determined. In appendix A, we show an example system to explain how to express and analyze a system by game tree.

## 3.2 Definition of Evenhanded System

Roughly speaking, an evenhanded situation means a situation where both parties have a strategy to receive a good (positive) payoff. For example, in the situation of rise of the value, the sender should have the choice of cancelling the session, and the receiver should have the choice to make the session valid. Therefore, we define an evenhanded situation as a period in which each participant has a strategy to obtain his/her positive payoff. However, a choice of moving into the next stage cannot be seen as such strategies. On an evenhanded stage, period of stay does not cause any disadvantage for each participant. In other words, all situations on the evenhanded stage are evenhanded situations; or, if one of the situations is disadvantageous for a party, then another situation is advantageous for the party.

---

<sup>2</sup> This rule is introduced because we assume that, if nothing happens, each participant wants to send/receive a mail by a faithful way.

From the above definitions, we say a system is evenhanded for contract with changeable values if all of the stages in the system are evenhanded.

### 3.3 Analysis Example

Fig.1 is an example of a typical on-line certified email system expressed by game tree. This system can be divided into two stages, and Fig.2 shows three situations of the second stage. Each square in Fig.2 denotes a termination of the session, and numbers in a tuple under each square denote payoffs for the participants. (Left number is for the sender, and right one is for the receiver.) In Fig.2,  $v_i$  means the value of the exchanged item on  $i$ th stage, and  $v_1$  is the initial value.

The faithful execution of the system is as follows. First, a sender sends an encrypted message (using the TTP's public key) and a hash value of the message to her intended receiver (denoted as procedure  $F_1$ ). Next, if the receiver wants to read the message, he sends the encrypted message and a signature of the hash value to the TTP (denoted as procedure  $F_2$ ). If the signature is valid, the TTP sends the decrypted message to the receiver and the receipt of the message to the sender, and this session can be completed faithfully. Also, the sender can perform a cancel protocol ( $C$ ) anytime before a termination of the session.

On the second stage of the example system, the sender has choices of  $C$  and  $S$  while the receiver has choices of  $F_2$  and  $S$ .

- In the case of  $v_1 > v_2$ , the sender's choice is  $S$  and the receiver's choice is  $S$ . This strategy results in payoffs of  $(-1, 1)$ .
- In the case of  $v_1 = v_2$ , the sender's choice is  $S$  and the receiver's choice is  $F_2$ . This strategy results in payoffs of  $(p_f, p_f)$ .
- In the case of  $v_1 < v_2$ , the sender's choice is  $C$  and the receiver's choice is  $F_2$ . This strategy results in payoffs of either  $(1, -1)$  or  $(p_f - 1, p_f + 1)$ .

From the above analysis, we can see that situations of " $v_1 < v_2$ " and " $v_1 = v_2$ " on the second stage are evenhanded because both parties have choices to gain a positive payoff. (In the case of  $v_1 < v_2$ , the payoffs depend on the result of

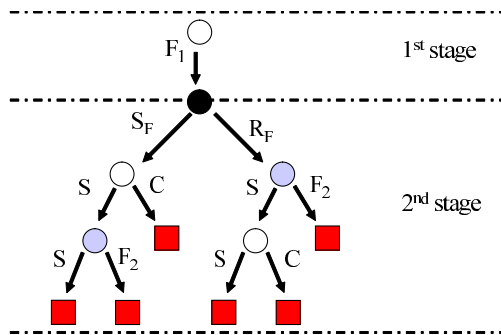


Fig. 1. Example System

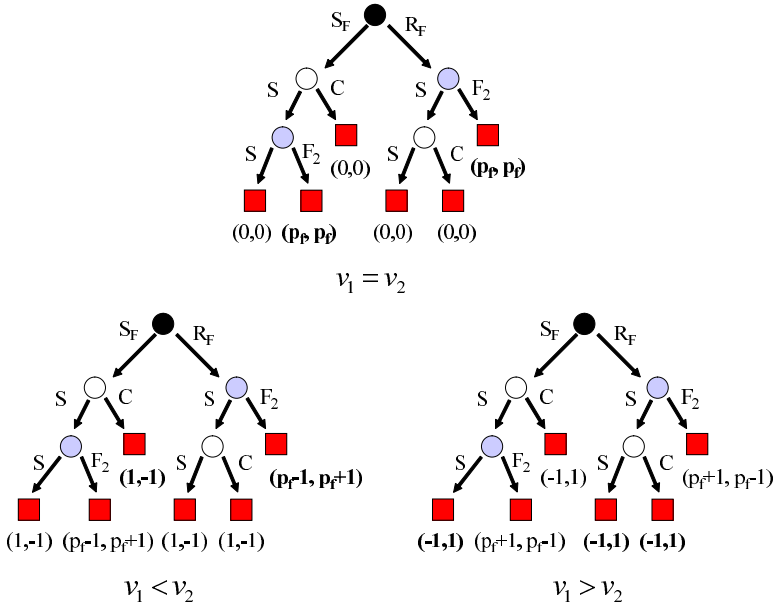


Fig. 2. Situations at 2nd Stage of Example System

the chance move). However, the situation of “ $v_1 > v_2$ ” is advantageous for the receiver because the receiver can always receive a positive payoff if he always selects a rational choice (his payoff must be 1) but the sender cannot (her maximum payoff is -1). Therefore, neither the second stage nor the whole system is evenhanded.

### 4 Analysis of Existing Systems

Here we express two existing systems [3, 1] by game tree, and demonstrate that they are not evenhanded systems as defined in Section 3. Due to the page limit, we omit the figures of their game trees.

#### System of Ateniese et al. [3]

On the second stage, the sender has no choice but  $S$  while the receiver has choices of  $F_2$ ,  $H$  and  $S$ .

- In the case of  $v_1 > v_2$ , the sender’s choice is  $S$  and the receiver’s choice is  $S$ . This strategy results in payoffs of  $(-1, 1)$
- In the case of  $v_1 < v_2$ , the sender’s choice is  $S$  and the receiver’s choice is  $F_2$ . This strategy results in payoffs of  $(p_f - 1, p_f + 1)$ .

From the above analysis, both situations on the second stage are advantageous for the receiver. Therefore, this system is not evenhanded (advantageous for the receiver on the second stage).

### System of Onieva et al. [1]

On the second stage, the sender has choices of  $C$  and  $S$  while the receiver has choices of  $F_2$ ,  $H$  and  $S$ .

- In the case of  $v_1 > v_2$ , the sender's choice is  $S$  and the receiver's choice is  $S$ . This strategy results in payoffs of  $(-1, 1)$ .
- In the case of  $v_1 < v_2$ , the sender's choice is  $C$  and the receiver's choice is  $H$ . This strategy results in payoffs of  $(1, -1)$  or  $(-1, 1)$ .

From the above analysis, the situation of  $v_1 > v_2$  is advantageous for the receiver, and the situation of  $v_1 < v_2$  is evenhanded. Therefore, this stage is not evenhanded (advantageous for the receiver in case of  $v_1 > v_2$  on the second stage).

On the third stage, the sender has choices of  $F_3$ ,  $C$  and  $S$  while the receiver has choices of  $H$  and  $S$ .

- In the case of  $v_1 > v_3$ , the sender's choice is  $F_3$  and the receiver's choice is  $S$ . This strategy results in payoffs of  $(p_f + 1, p_f - 1)$ .
- In the case of  $v_1 < v_3$ , the sender's choice is  $C$  and the receiver's choice is  $H$ . This strategy results in payoffs of  $(1, -1)$  or  $(-1, 1)$ .

From the above analysis, the situation of  $v_1 > v_3$  is advantageous for the sender, and the situation of  $v_1 < v_3$  is evenhanded. Therefore, this stage and whole system is not evenhanded (advantageous for the sender in case of  $v_1 > v_3$  on the third stage).

In addition to the above analysis, no existing system, as long as we have investigated [2, 4, 5, 6], is evenhanded. Especially, in any optimistic systems, which use an off-line TTP [1, 2, 4, 5], the receiver is advantageous on the second stage where  $H$  choice is available, but the sender is advantageous on the following stage(s) because no system prepares the cancel choice for the receiver. Hence, on the second stage, the rational receiver always selects the choice of  $H$  or  $S$  to prevent his disadvantageous stage because there is no reason for the receiver to execute faithfully. (Compared with the risk of meeting disadvantageous stage, the benefit of  $p_f$  might be not very attractive for the receiver.) As a consequence, for contract with changeable values, rational party uses a non-evenhanded optimistic system in the same way as on-line systems, in which the TTP is always used.

## 5 An Evenhanded Certified Email System

In order to prevent disadvantageous stage, an evenhanded system needs to provide "choice of cancel", for both the sender and the receiver. The reason why no previous system using an optimistic protocol provides the receiver with the choice of cancel is that the TTP does not have a way to check whether the receiver actually received the mail or not. Cancelling a session without checking

the receiver’s receipt can break fairness because the sender cannot make the receipt valid after cancelling the session while the receiver might have actually received the mail. To provide a way to check the receiver’s receipt, we introduce a *bulletin board* on which anyone can check all posted entries. In this section, we propose an evenhanded certified email system and analyze it in terms of security and actual management. Our proposed system needs some assumptions as follows.

- The bulletin board stores and publishes all authorized entries, and maintains the order of the posted entries.
- The TTP performs nothing except the procedures set by the system.
- The channel between the TTP and the bulletin board is authentic.
- Any party has his/her own public key pair, and knows the other party’s public key.

### 5.1 Proposed System

Our system consists of three protocols: *main protocol*, *help protocol*, and *cancel protocol*. Main protocol is used as a faithful execution to exchange the sender’s variable item and the receiver’s receipt, and the TTP does not appear. Help protocol can be initiated by the receiver before termination of a session, and used to make the session valid with the TTP’s help. Both parties can initiate cancel protocol before termination of a session, and it is used to cancel the session. Table 1 outlines the notation used in the protocol description.

#### Main Protocol (Faithful Execution)

- Start message ( $S \rightarrow R$ ):  $S2R, S2T, H(K), EO$
- Response message ( $R \rightarrow B$ ):  $SID, S2R, S2T, H(K), EO, ER,$   
 $MAC_P(S2R, S2T, H(K), EO, ER)$
- Finish message ( $S \rightarrow B$ ):  $SID, K, MAC_P(K)$

Before executing the main protocol, the sender shares  $SID$  (Session ID) and  $P$  (Password) with the bulletin board. Next, the sender randomly selects a session

**Table 1.** Notation

$S, R, T, B$	sender/ receiver/ TTP/ bulletin board
$H(X)$	a one-way cryptographic hash value of $X$
$PUB_X(Y)$	an encrypted message of $Y$ by $X$ ’s public key
$SIG_X(Y)$	a signature of $Y$ by $X$ ’s private key
<i>cleartext</i>	the name and price of the item
$S2R$	$PUB_R(SID, P, S, R, E_K(M), cleartext)$
$S2T$	$PUB_T(SID, S, R, K)$
$EO$	$SIG_S(SID, S2R, S2T, H(K))$
$ER$	$SIG_R(EO)$
$R2T$	$SIG_R(S2T, help)$

key  $K$ , and sends *Start message* to the intended receiver <sup>3</sup>. *cleartext* includes the description of the exchanged item ( $=M$ ) and its initial value ( $=v_1$ ).

After receiving the start message, the receiver verifies the signature of  $EO$ . If it is invalid, then abort. Otherwise, the receiver decrypts  $S2R$  with his private key and obtains  $SID$ ,  $P$ , and  $E_K(M)$ . If *Cancel message* has not been published on the bulletin board, the receiver sends *Response message* to the bulletin board. Then, the bulletin board checks the validity of  $MAC_P(S2R, S2T, H(K), EO, ER)$  included in the posted message, and if it is valid,  $S2R$ ,  $S2T$ ,  $H(K)$ ,  $EO$ , and  $ER$  are published on the board. Otherwise, the message is rejected.

Next, the sender verifies the signature of posted  $ER$  with the receiver's public key. If it is invalid, then abort. Otherwise, if neither *Finish message* nor *Help message* has been posted, the sender sends *Finish message* to the bulletin board. Then, the bulletin board checks the validity of  $MAC_P(K)$  included in the posted message, and if it is valid,  $K$  is published on the board. Otherwise, the message is rejected.

Finally, the receiver can receive the message by decrypting  $E_K(M)$  with the published  $K$ . On the other hand, the sender can insist on the receiver's receipt of  $M$  by showing the posted entries on the board.

### Help Protocol

- Request message ( $R \rightarrow T$ ):  $R2T$
- Help message ( $T \rightarrow B$ ):  $K$

This protocol can be initiated by the receiver if neither valid *Finish message* nor *Cancel message* has been posted on the bulletin board. First, the receiver signs  $S2T$  included in *Start message*, and sends *Request message* to the TTP.

Next, the TTP checks whether valid *Response message* has been posted on the bulletin board or not, and if it is published, the TTP decrypts  $S2T$  with its private key, and publishes  $K$  included in it.

After all, the receiver can receive the message by decrypting  $E_K(M)$  with the published  $K$  and the sender can insist on the receiver's receipt of  $M$  by showing the posted entries on the board.

### Cancel Protocol

- Cancel message ( $S$  or  $R \rightarrow B$ ):  $SID, cancel, MAC_P(cancel)$

This protocol can be initiated by either the sender or the receiver if neither valid *Finish message* nor *Help message* has been posted on the bulletin board. The bulletin board checks the validity of  $MAC_P(cancel)$  included in the posted message, and if it is valid, the cancel message is published on the board. Otherwise, the message is rejected. By completing this protocol, the session can be

---

<sup>3</sup> In the case that  $M$  is a big message to be exchanged, to improve the efficiency, a session key  $L$  may be introduced and  $S2R$  can be re-defined as  $S2R = E_L(E_K(M)), PUB_R(SID, P, S, R, L, cleartext)$ .

cancelled. However, if valid *Finish message* or *Help message* has been posted before *Cancel message*, the cancel is invalid.

If a session is cancelled, the bulletin board stops receiving new entries of the session. Without this rule, an unfair situation can happen in a race condition: the sender sends *Finish message* to the bulletin board while the receiver sends *Cancel message* at the same time. In this case, the cancel becomes invalid but the receiver can get the message by decrypting  $E_K(M)$  with the published  $K$ .

## 5.2 Analysis

Now, we conduct an informal analysis of the proposed system against the requirements introduced in Section 2. In the proposed system, the sender's authentication and non-repudiation of origin are available by the verification of  $EO$ . Similarly, the receiver's authentication and non-repudiation of receipt are available by the verification of  $ER$ . The receiver cannot obtain  $M$  without knowing  $K$ , and the sender cannot make the receipt valid without publishing  $K$ . Moreover, a receipt cannot be cancelled after publishing  $K$ . These properties result in fairness of the whole system. In addition, timeliness is available by using the help protocol or the cancel protocol.

Since our system uses a bulletin board, actual management of the board should be considered. We especially consider *denial of service* (DoS) attack against the bulletin board as an important problem. In our system, because only parties who know the correct pair of  $SID/P$  are allowed to post on the board, plenty of waste posting from DoS attackers can be prevented. Moreover, by introduction of a unique  $SID$ , the board can detect DoS attacks that re-send previously transferred messages, and the receiver can also detect the replay attack which aims to make him believe the sender sends the message twice.

We also consider a case in which a wrong key  $K$  or  $S2T$  different from the one included in *Start message* is posted. To confirm the correctness of  $K$  in *Finish message*, B will check whether the hash value of  $K$  is equal to  $H(K)$  posted on the bulletin board. To confirm the correctness of  $S2T$  in *Request message*, the TTP will check whether  $S2T$  is equal to the one posted on the bulletin board. Moreover, the correctness of the posted  $S2T$  and  $H(K)$  can be confirmed by verification of  $EO$  which is also on the bulletin board.

If  $SID$  is generated at random, it is possible that the receiver uses this  $SID/P$  pair to initiate another session with another party. This problem can be solved by introducing both participants' identities into each  $SID$ . Other parties cannot use a  $SID$  with incorrect identities to post messages. As the sender (S) and the bulletin board (B) share  $SID$  and  $P$  before executing the main protocol, and  $SID$  and  $P$  will be different for each session, there should be an efficient mechanism to allow S and B to share  $SID$  and  $P$  before each session. Suppose S and B share a secret ( $P_{master}$ ) in advance. Then,  $SID$  and  $P$  for each session can be derived as  $SID = S||R||i$ , and  $P = H(S, R, i, P_{master})$ , where  $||$  means a concatenation and  $i$  is a time-stamp or a counter.

Compared with other systems' public-key operations (i.e., encryption, decryption, signature generation and verification), the computational complexity of a

normal execution in our system is low: the total numbers in [1, 2, 3, 4, 5] and ours are 6, 6, 9, 10, 7 and 6, respectively.

Next, we show that the proposed system is evenhanded (see Fig.3).

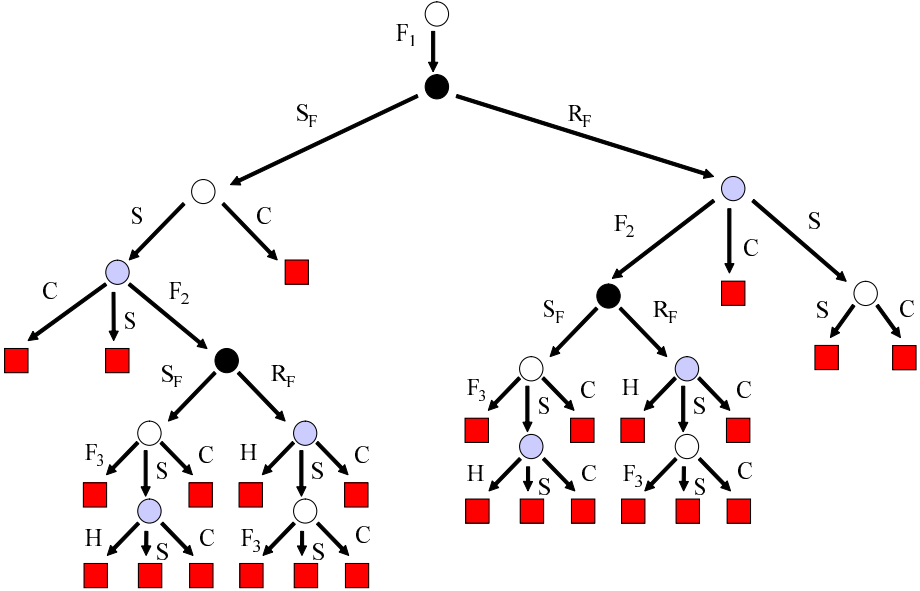


Fig. 3. Proposed System

– On the second stage, the sender has choices of  $C$  and  $S$  while the receiver has choices of  $F_2$ ,  $C$  and  $S$ .

- In the case of  $v_1 > v_2$ , the sender's choice is  $S$  and the receiver's choice is  $C$ . This strategy results in payoffs of  $(-1, 1)$ .
- In the case of  $v_1 < v_2$ , the sender's choice is  $C$  and the receiver's choice is  $F_2$ . This strategy results in payoffs of  $(1, -1)$  or it moves into the next stage.

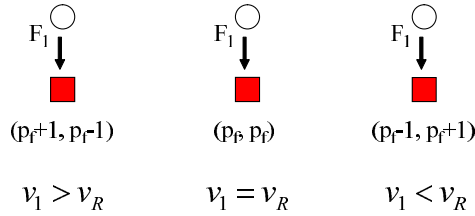
From the above analysis, the situation of  $v_1 > v_2$  is advantageous for the receiver, and the situation of  $v_1 < v_2$  is advantageous for the sender. Therefore, this stage is evenhanded.

– On the third stage, the sender has choices of  $F_3$ ,  $C$  and  $S$  while the receiver has choices of  $H$ ,  $C$  and  $S$ .

- In the case of  $v_1 > v_3$ , the sender's choice is  $F_3$  and the receiver's choice is  $C$ . This strategy results in payoffs of  $(p_f + 1, p_f - 1)$  or  $(-1, 1)$ .
- In the case of  $v_1 < v_3$ , the sender's choice is  $C$  and the receiver's choice is  $H$ . This strategy results in payoffs of  $(1, -1)$  or  $(-1, 1)$ .

From the above analysis, both situations at this stage are evenhanded.





**Fig. 4.** Situations of Toy Example

Since all stages in this system are evenhanded, it is an evenhanded certified email system.

Note, it is easy to design an evenhanded fair exchange protocol using the bulletin board without TTP. For example, sending all messages on the bulletin board, the posted message is used as the receiver's receipt of the message. However, considering exchange of items with changeable values, the assumption about adequate price offered at the starting point is not realistic. Then the above protocol without TTP is not evenhanded (see appendix A). For this reason, a contract of changeable values needs both participants' operations to make the contract evenhanded even if the assumption does not exist. In this case, even if using the bulletin board, it is not straightforward to design an evenhanded fair exchange protocol without TTP. This is because if the proposed protocol does not use TTP, the receiver loses the way to decrypt the message without the sender's help (i.e., the help choice to TTP  $H$  is not available), and the whole system becomes un-evenhanded. To design an evenhanded fair exchange protocol without TTP is one of our future works.

## 6 Conclusion

This paper considered a situation where a sender or a receiver can change his/her mind anytime before termination of a session. To cope with such a situation, we defined a notion of an evenhanded system by game tree and showed no previous system is evenhanded. We further proposed an evenhanded certified email system by using a bulletin board.

As far as we know, no existing system is evenhanded. So, it would be interesting to investigate how other non-evenhanded protocols can be turned into evenhanded ones.

## References

1. J. A. Onieva, J. Zhou, and J. Lopez, "Enhancing Certified Email Service for Timeliness and Multicasting", INC'04.
2. S. Kremer, and O. Markowitch, "Selective Receipt in Certified email", INDOCRYPT'01.

3. G. Ateniese, B. d. Medeiros, and M. T. Goodrich, "TRICERT: A Distributed Certified email Scheme", NDSS'01.
4. O. Markowitch and S. Kremer, "An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party," ISC'01.
5. G. Ateniese and C. N. Rotaru, "Stateless-Recipient Certified Email System based on Verifiable Encryption," CT-RSA'02.
6. M. Abadi, N. Glew, B. Horne and B. Pinkas, "Certified Email with a Light On-line Trusted Third Party: Design and Implementation," WWW'02.
7. R. Markle, "Secure Communications over Insecure Channels", Communications of the ACM 21:294-299, 1978.
8. L. Buttyan, and J-P. Hubaux, "Toward a Formal Model of Fair Exchange - a Game Theoretic Approach", EPFL SSC Technical Report No. SSC/1999/039.

## A Assumption About Adequate Initial Price Offered by the Sender

Under the model introduced in Section 2, we can easily design a simpler evenhanded system by using an on-line TTP. However, in such a system, if the assumption about adequate price offered at the starting point is not realized, some disadvantageous situation might happen.

Now we show a toy example which leads to a disadvantageous situation, and introduce requirements of contract systems against the problem. The procedure of the toy example is as follows. First, the sender sends  $S, R, M, cleartext, SIG_S(S, R, M, cleartext)$  to the TTP. Then, the TTP sends  $M$  to the intended receiver, and its receipt,  $SIG_T(SIG_S(S, R, M, cleartext))$ , to the sender. Because each session is completed soon after the initiation, the value of  $M$  does not change from the initial value  $v_1$ . Hence, this system is evenhanded, in which the payoffs are always  $(p_f, p_f)$ .

This system can be seen as an evenhanded system only based on the assumption that  $v_1$  offered by the sender is adequate. Otherwise, a problem will arise. Here,  $v_S$  ( $v_R$ ) denotes the adequate price for the sender (the receiver). At this time, this system has three situations for the receiver:  $v_1 > v_R$ ,  $v_1 = v_R$ , and  $v_1 < v_R$ . (The assumption means  $v_1 = v_R$ .) Regarding the case where the sender (receiver) sells (buys) the item at the lower (higher) price than the adequate price as a negative result, the payoffs in each situation are shown as in Fig.4 (suppose  $v_S = v_R$ ).

In this system, the sender can decide  $v_1$  and complete the exchange without the receiver's operation. As a natural result of this, a rational sender always decides  $v_1$ , where  $v_1 > v_S = v_R$ . This strategy results in the payoffs always being  $(p_f + 1, p_f - 1)$ . That is, in case the assumption  $v_1 = v_R$  is not realized, the sender is advantageous in this system.

For this reason, in a contract of changeable values, any session requires both participants' operations to make the contract valid as our proposed system introduced in Section 5. In addition, the conditions of a contract should be explicit such as the usage of *cleartext* to make both participants agree on the contract. In a system with these properties, each participant can choose to cancel a session if the offered value is not adequate.

# Efficient ID-Based Optimistic Fair Exchange with Provable Security

Zhenfeng Zhang<sup>1,2</sup>, Dengguo Feng<sup>1,2</sup>, Jing Xu<sup>1,3</sup>, and Yongbin Zhou<sup>1,2</sup>

<sup>1</sup> State Key Laboratory of Information Security

<sup>2</sup> Institute of Software, Chinese Academy of Sciences, Beijing 100080, P.R. China

<sup>3</sup> Graduate School of Chinese Academy of Sciences, Beijing 100039, P.R. China

zfzhang@is.iscas.ac.cn

**Abstract.** The notion of identity based cryptosystem was introduced by Shamir in 1984, and has attracted much interest since it eliminates the need of certificates and simplify the key management. In this paper, we propose an optimistic fair exchange protocol for identity-based signatures. A semi-trust third party (TTP) is still involved in our protocol to ensure fairness. However, there is no need for registrations between users and TTP, and no zero-knowledge proof is needed to provide verifiability. The proposed optimistic fair exchange protocol is much concise and efficient, and can be shown to be secure in the random model with a tight security reduction.

**Keywords:** Fair exchange, Identity-based Signature, Provable Security.

## 1 Introduction

With the growth of open networks such as Internet, the problem of fair exchanges has become one of the fundamental problems in secure electronic transactions and digital rights management. Payment systems, contract signing, electronic commerce and certified e-mail are classical examples in which fairness is a relevant security property. Informally, an exchange protocol allows two distributed parties to exchange electronic data in an efficient and fair manner, and it is said to be fair if it ensures that during the exchange of items, no party involved in the protocol can gain a significant advantage over the other party, even if the protocol is halted for any reason.

Protocols for fair exchange have attracted much attention in the cryptographic community in the past few years. The proposed methods mainly include: simultaneous secret exchange, gradual secret releasing, fair exchange using an on-line TTP and fair exchange with an off-line TTP. Among these results, optimistic fair exchange protocols based on an off-line trusted third party [1,5] are preferable as they offer a more cost-effective use of a trusted third party. An optimistic fair exchange protocol usually involves three parties: users Alice and Bob, as well as an off-line TTP. The off-line TTP does not participate the actual exchange protocol in normal cases, and is invoked only in abnormal cases to dispute the arguments between Alice and Bob to ensure fairness.

Asokan et al. [1] were the first to formally study the problem of optimistic fair exchanges. They present several provably secure but highly interactive solutions, based on the concept of *verifiable encryption of signatures*. Their approach was later generalized by [11], but all these schemes involve expensive and highly interactive zero-knowledge proofs in the exchange phase. Other less formal works on interactive verifiably encrypted signatures include [5,2]. The first and only non-interactive verifiably encrypted signature scheme was constructed by Boneh et al. [8], which is provably secure in the random oracle model, and is the first elegant scheme without a special registration between users and TTP and without zero-knowledge proofs.

Recently, Park et al. [17] proposed an optimistic protocol for fair exchange based on RSA signatures, using a technique of “two-signatures”. However, Park’s scheme was soon shown to be totally breakable in the registration phase by [14]. Moreover, Dodis and Reyzin [14] proposed a new primitive called *verifiably committed signatures* for constructing fair exchange protocols, and presented a committed signature scheme based on GDH signatures [9]. However, a registration protocol between TTP and users is still needed, and a zero-knowledge proofs of the equality of two discrete logarithms are involved to ensure the fairness.

Motivated by the approaches of verifiably encrypted signatures and verifiably committed signatures, the authors of [21] introduce a paradigm called *verifiable probabilistic signature schemes*, in which the exchanged items are probabilistic signatures. As probabilistic signatures has been studied extensively, this method seems rather natural. In their paradigm, a semi-trusted off-line TTP generates a trapdoor permutation as the system parameter, which can be used to produce verifiable partial signatures, while no further registrations between TTP and users is needed and no zero-knowledge proofs are involved. Thus, their framework has almost-optimal structures as that of [8]. Their approach is generic, and the resulting fair exchange protocol works especially with standard RSA signatures [18]. While being very concise and efficient, the only presented trapdoor permutation is based the factoring problem.

In 1984, Shamir [19] introduced the notion of identity-based cryptography (ID-PKC), in which a user’s public-key can be derived from his unique identifier information. The ID-PKC eliminates certificates and greatly simplifies the key management. A breakthrough work in the research of ID-PKC shall owe to Boneh and Franklin [10], who proposed the first efficient identity encryption scheme based on bilinear pairings over elliptic curves. Since then, a great deal of research has been done about the ID-based cryptosystems and protocols. However, as far as we know, no efficient identity based fair exchanges has been proposed. Although the approach proposed by Zhang et al. [21] can be applied to identity based signatures, the specified trapdoor permutation based on factoring may be not desirable in some applications, especially for schemes over elliptic curves.

In this paper, we propose an optimistic protocol for fair exchanges of identity-based signatures. A semi-trusted off-line TTP is still involved, who generates a public-key as the system parameter, while keeps the corresponding private-key secret to settle the dispute. No registration between users and TTP is needed and

no zero-knowledge proofs are involved. The underlying identity-based signatures utilizing bilinear pairings over elliptic curves, and the public-key chosen by TTP is a point over elliptic curves. The proposed protocol is as concise as that in [8,21], and is provably secure in the random oracle model.

It should be noted that, Micali [16] presented a fair electronic exchange protocol for contract signing with an invisible trusted party in PODC 2003, which has a similar framework that does not need registrations between users and TTP. However, Bao et al. [4] showed that Micali’s protocol cannot achieve the claimed fairness: the trusted party may face a dilemma situation that no matter what it does, one of the exchanging parties can succeed in cheating, and proposed a revised version that preserves fairness while remaining optimistic. Although the correctness of the revised version is believable, no security proof is provided. Moreover, a semantically secure encryption scheme under adaptive chosen ciphertext attacks is needed both in [16] and [4], and a random number specified by the initial party will be used by both parties for the underlying semantically-secure encryption.

The rest of the paper is organized as follows. In Section 2, we present a brief description of the formal model and security of identity-based verifiable probabilistic signatures. In Section 3, we propose a concrete identity-based verifiable probabilistic signature scheme and prove its security in the random oracle model. An identity-based optimistic fair exchange protocol is presented in section 4. Section 5 concludes the paper.

## 2 Verifiable Probabilistic Signature Model

Formal definitions of non-interactive fair exchanges via verifiable committed signatures or verifiable probabilistic signatures are proposed in [14] and [21] respectively, which explicitly consider the attack model and security goals, and result in a concrete description for the security against all parties involved in the protocols. Similar to [14,21], we present a formal model for the identity-based fair exchanges. And, to emphasize the role of *probabilistic* signature, we term it identity-based verifiable probabilistic signature scheme.

### 2.1 Definitions of ID-Based Verifiable Probabilistic Signatures

In an identity-based cryptosystem, there is a trusted authority called the *private key generator* (PKG) who holds a master key and issues private keys for all users in the system domain, and the public-key of a user can be derived publicly and directly from his unique identifier information. An identity-based verifiable probabilistic signature scheme involves three entities: a signer, a verifier and an arbitrator TTP, and is given by the following procedures.

**Setup:** System parameters  $\text{param}$  and a master key  $s$  is first generated by the PKG of an identity-based cryptosystem. A trapdoor one-way permutation is also published by a trusted third party (TTP) as a system parameter, that is, TTP

generates a key pair  $(PK, SK)$ , and makes  $PK$  public while keeps the trapdoor  $SK$  secret. Note that the TTP may be different from PKG.

**Extract:** Given a user's identity  $id$ , the PKG computes a private-key  $sk$  corresponding to  $id$  using his master-key  $s$ , and transmits it to the user. The user's public-key can be regarded as  $id$  and the corresponding private-key is  $sk$ .

**Psig and Pver:** These are probabilistic signing algorithm and verification algorithm. Given a message  $m$ , and private key  $sk$ , a signer outputs a probabilistic signature  $\sigma = \text{Psig}(sk, m)$ . The corresponding verification algorithm  $\text{Pver}(m, \sigma, id)$  takes as input  $m, \sigma$  and the signer's identity  $id$ , outputs 1 or 0.

**VPsig and VPver:** These are verifiable probabilistic signing and verification algorithms, which are just like an ordinary probabilistic signing and verification algorithms, except they depend on the public key  $PK$ . Given a message  $m$ , and keys  $sk$  and  $PK$ , a signer outputs a verifiable partial signature  $\sigma' = \text{VPsig}(sk, PK, m)$ . The corresponding verification algorithm  $\text{VPver}(m, \sigma', id, PK)$  takes as input  $m, \sigma'$  and public keys  $id$  and  $PK$ , outputs 1 (accept) or 0 (reject).

**Resolution Algorithm:** This is an algorithm run by an arbitrator TTP in case a singer refuses to open her probabilistic signature  $\sigma$  to a verifier, who in turn possesses a valid verifiable partial signature  $\sigma'$ . In this case,  $\text{Res}(m, \sigma', id, SK)$  should output a legal probabilistic signature  $\sigma$  on  $m$  of a signer with identity  $id$ .

The correctness of a verifiable probabilistic signature scheme states that

$$\begin{aligned} \text{Pver}(m, \text{Psig}(sk, m), id) &= 1, \\ \text{VPver}(m, \text{VPsig}(sk, PK, m), id, PK) &= 1, \\ \text{Pver}(m, \text{Res}(m, \sigma', id, SK), pk) &= 1. \end{aligned}$$

In a verifiable probabilistic signature model, TTP does not need to store anything except the trapdoor of the published one-way permutation. No further registration between users and TTP is needed, which will greatly reduce the communication overhead and managing cost. While in a verifiable committed signature scheme [14] and most of the verifiable encrypted signature schemes except [8], TTP shall maintain a secret-public key pair for each user via a registration phase, and the secret keys will then be used to resolve a dispute.

## 2.2 Security of ID-Based Verifiable Probabilistic Signatures

The security of a verifiable probabilistic signature scheme consists of ensuring fairness from three aspects: security against signer, security against verifier, and security against arbitrator. In the following, we denote by  $O_{\text{VPsig}}$  an oracle simulating the verifiable probabilistic signing procedure, and  $O_{\text{Res}}$  an oracle simulating the resolution procedure, and let  $O_{\text{Ext}}$  be an oracle simulating the private-key extracting operation. Let  $k$  be a suitable security parameter, and PPT stand for "probabilistic polynomial time".

**Security against a signer:** Intuitively, a signer should not be able to produce a verifiable probabilistic signature which is valid from a verifier's point of view, but which will not be extracted into a probabilistic signature of the signer

by an honest arbitrator TTP. More precisely, we require that any PPT adversary  $\mathcal{A}$  succeeds with at most negligible probability in the following experiment.

$$\begin{aligned} \text{Setup}(1^k) &\rightarrow (\text{param}, SK, PK) \\ (m, \sigma', id) &\leftarrow \mathcal{A}^{O_{\text{Res}}, O_{\text{Ext}}}(\text{param}, PK) \\ \sigma &\leftarrow \text{Res}(m, \sigma', id, SK) \\ \text{Success of } \mathcal{A} &= [\text{VPver}(m, \sigma', id, PK) = 1, \text{Pver}(m, \sigma, id) = 0]. \end{aligned}$$

In the model of considering security against signers, we allow an adversary  $\mathcal{A}$  to have the strongest power of extracting the private-key for any identity  $id$ .

**Security against a verifier:** A verifier should not be able to transfer any of the verifiable probabilistic signatures  $\sigma'$  that he got from a signer into a probabilistic signature  $\sigma$ , without explicitly asking TTP to do that. More precisely, any PPT adversary  $\mathcal{A}$  shall succeed with at most negligible probability in the following experiment:

$$\begin{aligned} \text{Setup}(1^k) &\rightarrow (\text{param}, SK, PK) \\ (m, \sigma, id) &\leftarrow \mathcal{A}^{O_{\text{VPSig}}, O_{\text{Res}}, O_{\text{Ext}}}(\text{param}, PK) \\ \text{Success of } \mathcal{A} &= [\text{Pver}(m, \sigma, id) = 1, m \notin \text{Query}(\mathcal{A}, O_{\text{Res}}), id \notin \text{Query}(\mathcal{A}, O_{\text{Ext}})], \end{aligned}$$

where  $\text{Query}(\mathcal{A}, O_{\text{Res}})$  is the set of valid queries  $\mathcal{A}$  asked to  $O_{\text{Res}}$ , i.e., the set of  $(m, \sigma', id)$  the adversary  $\mathcal{A}$  queried to  $O_{\text{Res}}$  satisfying  $\text{VPver}(m, \sigma', id, PK) = 1$ ,  $\text{Query}(\mathcal{A}, O_{\text{Ext}})$  is the set of queries  $\mathcal{A}$  asked to the private-key-extractor  $O_{\text{Ext}}$ .

**Security against the arbitrator:** An arbitrator's work is to check the validity of a request and recover the required probabilistic signature in case of dispute. However, a signer does not want the arbitrator to produce a valid probabilistic signature which she did not intend to produce, so we require the arbitrator to be *semi-trusted* in our model. To achieve this goal, we require that any PPT adversary  $\mathcal{A}$ , associated with verifiable probabilistic signing oracle  $O_{\text{VPSig}}$ , succeeds with at most negligible probability in the following experiment:

$$\begin{aligned} \text{Setup}^*(1^k) &\rightarrow (\text{param}, SK^*, PK) \\ (m, \sigma, id) &\leftarrow \mathcal{A}^{O_{\text{VPSig}}}(SK^*, \text{param}, PK) \\ \text{Success of } \mathcal{A} &= [\text{Pver}(m, \sigma, id) = 1, m \notin \text{Query}(\mathcal{A}, O_{\text{VPSig}})], \end{aligned}$$

where  $\text{Setup}^*(1^k)$  denotes the run of Setup with the dishonest arbitrator  $\mathcal{A}$ , and  $SK^*$  is her state after this run, and  $\text{Query}(\mathcal{A}, O_{\text{VPSig}})$  is the set of queries  $\mathcal{A}$  asked to the verifiable probabilistic signing oracle  $O_{\text{VPSig}}$ .

**Definition 1.** *A verifiable probabilistic signature scheme is secure if it is secure against signer's attack, verifier's attack and arbitrator's attack.*

### 3 ID-Based Verifiable Probabilistic Signature Scheme

We shall present a verifiable probabilistic signature scheme based on Bellare et al.s [7] modified Sakai-Ogishi-Kasahara signature scheme [20], which was com-

monly called SOK-IBS (for Sakai-Ogishi-Kasahara Identity Based Signature) in literatures [7]. In fact, the SOK-IBS scheme can be regarded as an identity based extension of a randomized version of Boneh et al.'s short signature scheme [9].

### 3.1 The Bilinear Pairing

Let  $\mathcal{G}_1$  be a cyclic additive group generated by  $P$ , whose order is a prime  $q$ , and  $\mathcal{G}_2$  be a cyclic multiplicative group of the same order. Let  $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$  be a pairing which satisfies the following conditions:

1. Bilinearity: For any  $P, Q, R \in \mathcal{G}_1$ , we have  $e(P + Q, R) = e(P, R)e(Q, R)$  and  $e(P, Q + R) = e(P, Q)e(P, R)$ . In particular, for any  $a, b \in \mathbf{Z}_q$ ,

$$e(aP, bP) = e(P, P)^{ab} = e(P, abP) = e(abP, P).$$

2. Non-degeneracy: There exists  $P, Q \in \mathcal{G}_1$ , such that  $e(P, Q) \neq 1$ .

3. Computability: There is an efficient algorithm to compute  $e(P, Q)$  for all  $P, Q \in \mathcal{G}_1$ .

The typical way of obtaining such pairings is by deriving them from the Weil-pairing or the Tate-pairing on an elliptic curve over a finite field. We refer to [9,10] for a more comprehensive description on how these groups, pairings and other parameters should be selected for efficiency and security.

Computation Diffie-Hellman (CDH) Problem: Given  $P, aP, bP \in \mathcal{G}_1$  for randomly chosen  $a, b \in_{\mathcal{R}} \mathbf{Z}_q^*$ , to compute  $abP$ .

### 3.2 The Proposed Scheme

Since 2001, all kinds of identity-based cryptosystems have been proposed based on the bilinear maps, such as Weil-pairing or Tate-pairing on an elliptic curve over a finite field. The following is a brief overview of the identity-based setting. We refer to [10] for a detailed description.

• **Setup:** Given a security parameter  $k$ , the PKG chooses groups  $\mathcal{G}_1$  and  $\mathcal{G}_2$  of prime order  $q > 2^k$ , a generator  $P$  of  $\mathcal{G}_1$ , a bilinear map  $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ , a randomly chosen master key  $s \in \mathbf{Z}_q^*$  and the associated public key  $P_{pub} = sP$ . It also picks cryptographic hash functions of same domain and range  $H_1, H_2 : \{0, 1\}^* \rightarrow \mathcal{G}_1$ . The system's public parameters are **params** =  $(\mathcal{G}_1, \mathcal{G}_2, e, P, P_{pub}, H_1, H_2)$ .

The TTP chooses  $x \in \mathbf{Z}_q^*$  at random, generates a public key  $PK = xP$  and publishes it as a system parameter, and keeps  $SK = x$  secret.

• **Extract:** Suppose the identity of a user is  $ID$ . Given an identity  $ID$ , the PKG computes  $Q_{ID} = H_1(ID) \in \mathcal{G}_1$  and  $d_{ID} = sQ_{ID} \in \mathcal{G}_1$  that is transmitted to the user. The user's private-key is  $d_{ID}$ , which satisfies  $e(d_{ID}, P) = e(Q_{ID}, P_{pub})$ .

• **Psig and Pver:** These are the signing and verification algorithms of the SOK-IBS scheme. In order to sign a message  $m$ , Psig perform as following:



- Pick  $r \in_{\mathcal{R}} \mathbf{Z}_q$ , compute  $U = rP \in \mathcal{G}_1$  and  $H = H_2(ID, m, U) \in \mathcal{G}_1$ .
- Compute  $V = d_{ID} + rH \in \mathcal{G}_1$ .

The signature on  $m$  is the pair  $\sigma = \langle U, V \rangle \in \mathcal{G}_1 \times \mathcal{G}_1$ .

To verify a signature  $\sigma = \langle U, V \rangle \in \mathcal{G}_1 \times \mathcal{G}_1$  on a message  $m$  for an identity  $ID$ , the algorithm  $\text{Pver}$  first takes

$$Q_{ID} = H_1(ID) \in \mathcal{G}_1 \quad \text{and} \quad H = H_2(ID, m, U) \in \mathcal{G}_1,$$

and then accepts the signature if and only if

$$e(P, V) = e(P_{pub}, Q_{ID}) \cdot e(U, H). \quad (1)$$

The signature scheme constituted of  $(\text{Psig}, \text{Pver})$  is actually the SOK-IBS scheme, which has been proved [7] to be non-existentially forgeable against adaptive chosen message attacks in the random oracle model [6], first by Bellare etc. [7] through a general framework applying to a large family of schemes, and then by Libert and Quisquater [15], who showed that SOK-IBS scheme has a much tighter security proof under the CDH assumption.

• **VPsig and VPver:** To generate a partial signature a message  $m$ ,  $\text{VPsig}$  performs as following:

- First choose  $r \in \mathbf{Z}_q$  at random and compute  $U = rP \in \mathcal{G}_1$ , and then let  $H = H_2(ID, m, U) \in \mathcal{G}_1$ .
- Compute  $V' = d_{ID} + rH + rPK \in \mathcal{G}_1$ .

The verifiable partial signature on  $m$  is the pair  $\sigma' = \langle U, V' \rangle \in \mathcal{G}_1 \times \mathcal{G}_1$ .

The corresponding verification algorithm  $\text{VPver}$  takes as input  $\sigma'$ ,  $ID$  and  $PK$ . It first computes  $Q_{ID} = H_1(ID) \in \mathcal{G}_1$  and  $H = H_2(ID, m, U) \in \mathcal{G}_1$ , and then accepts the signature if

$$e(P, V') = e(P_{pub}, Q_{ID}) \cdot e(U, H + PK), \quad (2)$$

and rejects it otherwise.

• **Res:** Given a verifiable partial signature  $\sigma' = (U, V')$  on a message  $m$  for an identity  $ID$ , the arbitrator TTP first verifies its validity by checking (2). If valid, TTP computes

$$V = V' - xU \quad (3)$$

and returns  $\sigma = (U, V) = \text{Res}(m, \sigma', ID, SK)$  as a probabilistic signature of  $m$  to the verifier.

Note that, TTP holds the trapdoor  $SK = x$ , and if  $U = rP$ , then we have

$$r \cdot PK = r \cdot xP = x \cdot rP = xU.$$

That is, if  $U = rP$  and  $V' = d_{ID} + rH + rPK$ , then  $V = V' - xU = d_{ID} + rH$ . Thus  $\langle U, V \rangle$  is a regular SOK-IBS signature on message  $m$  for the identity  $ID$ .

**Remark:** The following facts shall be noted.

- Similar to the proofs [7,15] of SOK-IBS signature scheme, the verifiable partial signature scheme constituted of (VPsig, VPver) can be shown to be non-existential forgeable under adaptive chosen message attacks in the random oracle model, assuming the CDH problem in  $\mathcal{G}_1$  is hard.
- The signer’s identity  $ID$  is explicitly included in the signature as  $H = H_2(ID, m, U)$ , thus the colluding attacks proposed by Bao [3] will not work.
- This approach also works for other identity-based signatures, such as [12].
- In our scheme, the TTP is semi-trusted and can be any party different from PKG. This makes our scheme more flexible. In fact, if we designate PKG as the arbitrator, then this TTP must be fully trusted since every user’s private key is escrowed by it.

### 3.3 Security of Our Scheme

**Theorem 1.** *Under the formal model described in section 2, the verifiable probabilistic signature scheme based on SOK-IBS is provably secure in the random oracle model, provided that the CDH problem is hard.*

*Proof.* According to Definition 1, we shall show that the proposed verifiable probabilistic signatures is secure against signer, verifier and arbitrator. As we will see, the security against a signer follows unconditionally, and an arbitrator’s attack can be converted into a forger for the SOK-IBS signature, while a verifier’s attack can be related to the CDH problem. The major difficulty for the proof of security against a verifier comes from the dealing with the resolution queries.

**Security against signer’s attack:** With the help of the oracles  $O_{\text{Res}}$  and  $O_{\text{Ext}}$ , a malicious signer’s goal is to produce a valid verifiable partial signature  $\sigma' = (U, V')$ , which cannot be extracted into a valid probabilistic signature  $\sigma = (U, V)$ . However, this is always not the case. In fact, for any  $\sigma' = (U, V')$  satisfying  $e(P, V') = e(P_{\text{pub}}, Q_{ID}) \cdot e(U, H + PK)$ , and  $V = V' - xU$ , we have

$$e(P, V) = e(P, V')e(P, -xU) = e(P, V')e(PK, U)^{-1} = e(P_{\text{pub}}, Q_{ID}) \cdot e(U, H).$$

Thus the  $(U, V)$  extracted by TTP is definitely a valid SOK-IBS signature on  $m$ , and the signer Alice cannot deny it. In fact, the oracle  $O_{\text{Res}}$  cannot give any help to a malicious signer: what  $O_{\text{Res}}$  extracted is exactly the  $xU$ , which was already known to her as  $rPK = rXP = xU$ .

**Security against verifier’s attack:** Making use of the oracles  $O_{\text{VPsig}}$ ,  $O_{\text{Ext}}$  and  $O_{\text{Res}}$ , an adversarial verifier wins if he forges a valid probabilistic signature  $\sigma = (U, V)$  for an entity with identity  $ID$ , for which the corresponding verifiable partial signature  $\sigma' = (U, V')$  has not been queried to  $O_{\text{Res}}$ , and  $ID$  has not been queried to  $O_{\text{Ext}}$ . We shall show that such an attack can be used by a probabilistic polynomial time algorithm  $\mathcal{F}$  to solve the CDH problem.

Let  $(X = aP, Y = bP) \in \mathcal{G}_1 \times \mathcal{G}_1$  be a random instance of the CDH problem taken as input by  $\mathcal{F}$ .  $\mathcal{F}$  takes  $z \in \mathbf{Z}_q^*$  at random and sets  $PK = zY$ , and

then initializes Bob with  $P_{pub} = X$  and  $PK$  as system's overall public keys. The algorithm  $\mathcal{F}$  then starts performing queries such as those required by an identity based setting and the security model described as in section 2. Without loss of generality, we assume that, for any key extraction query or signature query involving an identity, a  $H_1$  oracle query was previously issued for the same identity. Then these queries are answered by  $\mathcal{F}$  as follows.

– *Queries on oracle  $H_1$* : When an identity  $ID$  is submitted to the  $H_1$  oracle, as in Coron's proof technique [13],  $\mathcal{F}$  flips a coin  $T \in \{0, 1\}$  that yields 0 with probability  $\delta$  and 1 with probability  $1 - \delta$ .  $\mathcal{F}$  then picks  $w \in \mathbf{Z}_q^*$ . If  $T = 0$  then the hash value  $H_1(ID)$  is defined as being  $wP \in \mathcal{G}_1$ . If  $T = 1$ , then  $\mathcal{F}$  returns  $wY \in \mathcal{G}_1$ . In both cases,  $\mathcal{F}$  inserts a tuple  $(ID, w, T)$  in a list  $L_1$  to keep track of the way it answered the query.

– *Key extraction queries*: When Bob requests the private key associated to an identity  $ID$ ,  $\mathcal{F}$  recovers the corresponding  $(ID, w, T)$  from  $L_1$  (recall that such a tuple must exist because of the aforementioned assumption). If  $T = 1$ , then  $\mathcal{F}$  outputs “failure” and halts because it is unable to coherently answer the query. Otherwise, it means that  $H_1(ID)$  was previously defined to be  $wP \in \mathcal{G}_1$  and  $wP_{pub} = wX$  is then returned to Bob as a private key associated to  $ID$ .

– *Queries on oracle  $H_2$* : When a tuple  $(ID, m, U)$  is submitted to the  $H_2$  oracle,  $\mathcal{F}$  first scans a list  $L_2$  to check whether  $H_2$  was already defined for that input. If it was, the previously defined value is returned. Otherwise,  $\mathcal{F}$  picks a random  $v \in \mathbf{Z}_q^*$ , stores the tuple  $(ID, m, U, v)$  in the list  $L_2$  and returns  $H = vP \in \mathcal{G}_1$  as a hash value to Bob.

– *Partial signature queries  $O_{VPsig}$* : When Bob queries the partial signature oracle  $O_{VPsig}$  on a message  $m_i$  for an identity  $ID$ ,  $\mathcal{F}$  first recovers the previously defined value  $Q_{ID} = H_1(ID)$  from  $L_1$ . (1) If  $Q_{ID} = wP$ ,  $\mathcal{F}$  randomly chooses  $v_i \in \mathbf{Z}_q^*$  and sets  $U_i = v_iP$  and  $V_i' = wP_{pub} + v_i(H + PK)$ . (2) If  $Q_{ID} = wY$ , it chooses numbers  $t_i, v_i \in \mathbf{Z}_q^*$  at random, and then sets  $V_i' = t_iP_{pub}$ ,  $U_i = v_iP_{pub}$ , and defines the hash value  $H_2(ID, m_i, U_i)$  as  $H = v_i^{-1}(t_iP - Q_{ID}) - PK \in \mathcal{G}_1$  ( $\mathcal{F}$  halts and outputs “failure” if  $H_2$  turns out to be already defined for the input  $(ID, m_i, U_i)$ ). The  $(U_i, V_i')$  is returned to Bob and appears as a valid verifiable partial signature from the latter's point of view, since

$$(1) e(P, V_i') = e(wP, P_{pub})e(v_iP, H + PK) = e(P_{pub}, Q_{ID})e(U_i, H + PK);$$

$$(2) e(P_{pub}, Q_{ID})e(U_i, H + PK) = e(P_{pub}, Q_{ID})e(v_iP_{pub}, v_i^{-1}(t_iP - Q_{ID})) \\ = e(P_{pub}, Q_{ID})e(P_{pub}, (t_iP - Q_{ID})) \\ = e(P_{pub}, t_iP) = e(t_iP_{pub}, P) = e(P, V_i').$$

$\mathcal{F}$  keeps a list of  $L_3 = \{(ID, m_i, U_i, V_i', v_i)\}$ .

– *Resolution queries  $O_{Res}$* : When Bob queries the resolution oracle  $O_{Res}$  on a partial signature  $(m, U, V')$  for an identity  $ID$ ,  $\mathcal{F}$  first check its validity and recovers the previously defined value  $Q_{ID} = H_1(ID)$  from  $L_1$ . If  $T = 1$ , it halts and outputs “failure”. Otherwise,  $\mathcal{F}$  looks up the list  $L_3$ , finds out  $v_i$  and answers

Bob with  $V = V' - v_i PK$  if  $(m, U, V')$  is in the list, and halts otherwise. Note that, if  $(U, V')$  is a valid partial signature and if  $V' = v_i P$ , then  $(U, V)$  is a valid SOK-IBS signature. And since the partial signature scheme  $(\text{VPsig}, \text{VPver})$  is non-existential forgeable under adaptive chosen message attacks, assuming the CDH problem is hard, the probability that  $m$  has not been queried to  $O_{\text{VPsig}}$  (which means that  $(m, U, V')$  is a valid forgery) is negligible, and so is it with  $\mathcal{F}$  halts in answering  $O_{\text{Res}}$ -queries if  $T = 0$ .

Suppose Bob outputs a fake signature  $\tilde{\sigma} = (\tilde{m}, \tilde{U}, \tilde{V})$  for an identity  $\tilde{ID}$  eventually.  $\mathcal{F}$  then recovers the triple  $(\tilde{ID}, \tilde{w}, \tilde{T})$  from  $L_1$ . If  $\tilde{T} = 0$ , then  $\mathcal{F}$  outputs “failure” and stops. Otherwise, it goes on and finds out whether  $(\tilde{ID}, \tilde{m}, \tilde{U}, \cdot, \cdot)$  appears in the list  $L_3$ . Suppose it does not appear in the list  $L_3$ , then the list  $L_2$  must contain an entry  $(\tilde{ID}, \tilde{m}, \tilde{U}, \tilde{v})$  with overwhelming probability (otherwise, B stops and outputs “failure”). Then, since  $\tilde{H} = H_2(\tilde{ID}, \tilde{m}, \tilde{U})$  was defined to be  $\tilde{v}P \in \mathcal{G}_1$ , if Bob succeeded in the game with the view it was provided with,  $\mathcal{F}$  knows that

$$e(P, \tilde{V}) = e(X, Q_{\tilde{ID}})e(\tilde{U}, \tilde{H})$$

with  $\tilde{H} = \tilde{v}P$  and  $Q_{\tilde{ID}} = \tilde{w}Y$  for known elements  $\tilde{w}, \tilde{v} \in \mathbf{Z}_q^*$ . Then, it is also known that

$$e(P, \tilde{V} - \tilde{v}\tilde{U}) = e(X, \tilde{w}Y),$$

and thus  $\tilde{w}^{-1}(\tilde{V} - \tilde{v}\tilde{U})$  is the solution to the CDH instance  $(X, Y) \in \mathcal{G}_1 \times \mathcal{G}_1$ .

If  $(\tilde{ID}, \tilde{m}, \tilde{U}, \cdot, \cdot)$  does appear in the list  $L_3$ , then  $(\tilde{ID}, \tilde{m}, \tilde{U}, \cdot)$  most not have been queried to the oracle  $O_{\text{Res}}$ .  $\mathcal{F}$  goes through the list  $L_3$  to find out the  $\tilde{v}$ , for which  $\tilde{U} = \tilde{v}P_{\text{pub}} = \tilde{v}X$ . Note that  $(\tilde{U}, \tilde{V})$  and  $(\tilde{U}, \tilde{V}')$  are verifiable partial signature and SOK-IBS signature on  $\tilde{m}$  respectively. From (1) and (2) we have

$$e(\tilde{V}' - \tilde{V}, P) = e(\tilde{U}, PK) = e(\tilde{v}X, zY) = e(X, Y)^{\tilde{v}z},$$

and thus  $(\tilde{v}z)^{-1}(\tilde{V}' - \tilde{V})$  is the solution to the CDH instance  $(X, Y) \in \mathcal{G}_1 \times \mathcal{G}_1$ .

Assume that a PPT verifier Bob has an advantage  $\varepsilon$  in forging a signature in an attack modelled by the game of section 2, when running in a suitable time and asking  $q_{H_i}$  queries to random oracles  $H_i (i = 1, 2)$ ,  $q_E$  queries to the key extraction oracle,  $q_S$  queries to the verifiable partial signature oracle, and  $q_{\text{Res}}$  queries to the resolution oracle.

When assessing  $\mathcal{F}$ 's probability of failure, one readily checks that its probability to fail in handling a signing query because of a conflict on  $H_2$  is at most  $q_S(q_{H_2} + q_S)/2^k$  (as  $L_2$  never contains more than  $q_{H_2} + q_S$  entries) while the probability for Bob to output a valid forgery  $(\tilde{U}, \tilde{V})$  on  $\tilde{M}$  without asking the corresponding  $H_2(\tilde{ID}, \tilde{m}, \tilde{U})$  query is at most  $1/2^k$ . Finally, by an analysis similar to Coron's one [13], the probability  $\delta^{q_E + q_{\text{Res}}}(1 - \delta)$  for  $\mathcal{F}$  not to fail in a key extraction query or a resolution query or because Bob produces its forgery on a ‘bad’ identity  $\tilde{ID}$  is greater than  $1 - 1/e(q_E + q_{\text{Res}} + 1)$  when the optimal probability  $\delta_{\text{opt}} = (q_E + q_{\text{Res}})/(q_E + q_{\text{Res}} + 1)$  is taken. Eventually, it comes that  $\mathcal{F}$ 's advantage in solving the CDH problem in  $\mathcal{G}_1$  is at least

$$\left( \varepsilon - (q_S(q_{H_2} + q_S) + 1) / 2^k \right) / e(q_E + q_{\text{Res}} + 1).$$

**Security against arbitrator’s attack:** Now we consider an adversarial TTP’s attack. Holding the trapdoor  $SK = x$ , TTP can extract any  $U$  and  $V'$  into a pair of  $(U, V)$  satisfying (1). We shall also show a reduction of converting an arbitrator’s attack into a valid forgery for the SOK-IBS signature scheme. A forger  $\mathcal{F}$  accepts  $ID$  and  $PK$  as input. The TTP holds  $(PK, SK)$  and has access to the  $O_{VPsig}$ -oracle and the random oracle  $H_2$ , and wins if he forges a SOK-IBS signature  $\tilde{\sigma} = (\tilde{m}, \tilde{U}, \tilde{V})$ , while  $\tilde{m}$  has not been queried to  $O_{VPsig}$ -oracle.

Here is how  $\mathcal{F}$  invokes TTP. For an  $O_{VPsig}$ -query on message  $m$ ,  $\mathcal{F}$  chooses  $t_i, v_i \in \mathbf{Z}_q^*$  at random, and then sets  $V'_i = t_i P_{pub} \in \mathcal{G}_1$ ,  $U_i = v_i P_{pub} \in \mathcal{G}_1$ , and defines the hash value  $H_2(ID, m_i, U_i)$  as  $H = v_i^{-1}(t_i P - Q_{ID}) - PK \in \mathcal{G}_1$  ( $\mathcal{F}$  halts and outputs “failure” if  $H_2$  turns out to be already defined for the input  $(ID, m_i, U_i)$ ). The  $(U_i, V'_i)$  is returned to Bob as a valid verifiable partial signature. When TTP outputs a forgery  $(\tilde{m}, \tilde{\sigma})$  as described above, where  $\tilde{m}$  has not been queried to  $O_{VPsig}$ ,  $\mathcal{F}$  just outputs  $(\tilde{m}, \sigma)$ . We see that the simulation is perfect, and  $\mathcal{F}$  succeeds in generating a valid forgery if TTP succeeds.

From a TTP’s point of view, a  $O_{VPsig}$ -oracle is essentially a SOK-IBS signing oracle, since she holds the trapdoor  $SK = x$ . Therefore, what TTP is trying to do is to forge a valid SOK-IBS signature under adaptive chosen message attacks. Her advantage is negligible as the SOK-IBS scheme is non-existential forgeable against adaptive chosen-message attacks, under the CDH-assumption.

The above arguments show that, our scheme is provably secure under the well-known CDH assumption, of course, in the random oracle model.  $\square$

## 4 ID-Based Optimistic Fair Exchanges

Now we present an optimistic fair exchange protocol based on the probabilistic signatures described as in section 3. The construction is similar to [8], [14], [21].

Assume Alice’s identity is  $ID_A$  and the private key is  $d_A = sQ_A = sH_1(ID_A)$ , and Bob’s identity is  $ID_B$  and his private key is  $d_B = sQ_B = sH_1(ID_B)$ . The public key of a TTP is  $PK = xP$  while the private key is  $SK = x$ .

1. Alice first randomly chooses  $r_A \in \mathbf{Z}_q^*$  and computes  $U_A = r_A P \in \mathcal{G}_1$  and  $H_A = H_2(ID_A, m, U_A) \in \mathcal{G}_1$ , then computes  $V' = d_A + r_A H_A + r_A PK \in \mathcal{G}_1$ . Alice sends a verifiable partial signature  $\sigma'_{Alice} = (m, U_A, V'_A)$  to Bob.

2. Bob first computes  $Q_A = H_1(ID_A) \in \mathcal{G}_1$  and  $H_A = H_2(ID_A, m, U_A) \in \mathcal{G}_1$ . He then checks

$$e(P, V'_A) = e(P_{pub}, Q_A) \cdot e(U_A, H_A + PK).$$

If it is valid, Bob randomly chooses  $r_B \in \mathbf{Z}_q^*$  and computes  $U_B = r_B P \in \mathcal{G}_1$  and  $H_B = H_2(ID_B, m, U_B) \in \mathcal{G}_1$ , and then computes  $V'_B = d_B + r_B H_B \in \mathcal{G}_1$ . Bob sends his signature  $\sigma_{Bob} = (m, U_B, V'_B)$  to Alice.

3. After receiving Bob’s signature  $\sigma_{Bob} = (m, U_B, V'_B)$ , Alice first computes  $Q_B = H_1(ID_B) \in \mathcal{G}_1$  and  $H_B = H_2(ID_B, m, U_B) \in \mathcal{G}_1$ , and then verifies

$$e(P, V'_B) = e(P_{pub}, Q_B) \cdot e(U_B, H_B).$$

If valid, she computes  $V_A = V'_A - r_A PK$  and sends  $\sigma_{Alice} = (m, U_A, V_A)$  to Bob.

4. If Bob does not receive anything in step 3, or if Alice's signature  $\sigma_{Alice}$  is invalid, then he sends the verifiable partial signature  $\sigma'_{Alice} = (m, U_A, V'_A)$  and his probabilistic signature  $\sigma_{Bob} = (m, U_B, V_B)$  to TTP. This protocol provides a vehicle for TTP to understand whether the protocol was correctly carried out. TTP first computes  $Q_A = H_1(ID_A)$ ,  $H_A = H_2(ID_A, m, U_A)$ , and  $Q_B = H_1(ID_B)$ ,  $H_B = H_2(ID_B, m, U_B)$ , and then checks

$$e(P, V_B) = e(P_{pub}, Q_B) \cdot e(U_B, H_B),$$

and

$$e(P, V'_A) = e(P_{pub}, Q_A) \cdot e(U_A, H_A + PK).$$

If both are valid, TTP extracts  $V_A = V'_A - xU_A$ , and sends  $\sigma_{Alice} = (m, U_A, V_A)$  to Bob and sends  $\sigma_{Bob} = (m, U_B, V_B)$  to Alice.

## 5 Conclusion

We propose an efficient and optimistic fair exchange protocol of identity-based signatures and give a security proof with tight reduction in the random model. Similar to [21], a semi-trust third party (TTP) is still involved in our protocol to ensure fairness, while it is not required to store any information except its private-key. There is no need for registrations between users and TTP, and no zero-knowledge proof is involved. This is the first identity-based optimistic fair exchange protocol with such a concise framework.

## Acknowledgement

The work is supported by National Natural Science Foundation of China under Granted No.60373039, and National Grand Fundamental Research Project of China under Granted No.G1999035802.

## References

1. N.Asokan, V.Shoup, M.Waidner. Optimistic fair exchange of digital signatures. Advances in Cryptology - EUROCRYPT'98, LNCS 1403, pages 591-606, Springer-Verlag, 1998; IEEE J. on Selected Areas in Communication, 18(4): 593-610, 2000.
2. G.Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. Sixth ACM Conference on Computer and Communication Security, pages 138-146. ACM, 1999; Verifiable encryption of digital signatures and applications, ACM Transactions on Information and System Security, Vol. 7, No. 1, pages 1-20, 2004.
3. F. Bao. Colluding Attacks to a Payment Protocol and Two Signature Exchange Schemes. In ASIACRYPT 2004, LNCS 3329, pages 417-429, Springer-Verlag, 2004.
4. F. Bao, G.L. Wang, J.Y. Zhou, H.F. Zhu. Analysis and Improvement of Micali's Fair Contract Signing Protocol, ACISP 2004, LNCS 3108, pp. 176-187, 2004.
5. F. Bao, R.H. Deng, W. Mao. Efficient and practical fair exchange protocols with off-line TTP. IEEE Symposium on Security and Privacy, pages 77-85, 1998.

6. M. Bellare and P. Rogaway: Random oracles are practical: a paradigm for designing efficient protocols. Proceedings of the First Annual Conference on Computer and Communications Security, ACM, 1993.
7. M. Bellare, C. Namprempe and G. Neven. Security Proofs for Identity-Based Identification and Signature Schemes, Advances in Cryptology-Eurocrypt'04, LNCS 3027, pages 268-286, Springer-Verlag, 2004.
8. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. Advances in Cryptology-EUROCRYPT 2003, LNCS 2656, pages 416-432. Springer-Verlag, 2003.
9. D. Boneh, B. Lynn, H. Shacham. Short signatures from the weil pairing. Advances in Cryptology-ASIACRYPT 2001, LNCS 2248, pp.514-532, Springer-Verlag, 2001.
10. D. Boneh, M. Franklin: Identity-based encryption from the Weil Pairing. In Crypto'2001, LNCS 2139, Springer-Verlag, pages 213-229, 2001.
11. J. Camenisch and I. B. Damgard. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. Advances in Cryptology-ASIACRYPT 2000, LNCS 1976, pages 331-345, Springer-Verlag, 2000.
12. J.C. Cha, J.H. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups, Proc. of PKC 2003. Springer-Verlag, LNCS 2567, pp.18-30, Springer, 2003.
13. J.S. Coron. On the exact security of Full Domain Hash. Advances in Cryptology-Crypto 2000, LNCS 1880, pp.229-235, Springer-Verlag, 2000.
14. Y. Dodis and L. Reyzin. Breaking and Repairing Optimistic Fair Exchange from PODC 2003. ACM Workshop on Digital Rights Management, pages 47-54, 2003.
15. B. Libert, J.-J. Quisquater. The Exact Security of an Identity Based Signature and its Applications, IACR Cryptology ePrint Archive, Report 2004/102, 2004.
16. S. Micali. Simple and fast optimistic protocols for fair electronic exchange. 2003 ACM Symposium on Principles of Distributed Computing, pages 12-19, 2003.
17. J. M. Park, E. Chong, H. Siegel, I. Ray. Constructing fair exchange protocols for E-commerce via distributed computation of RSA signatures. In 22<sup>th</sup> ACM Symp. on Principles of Distributed Computing, pages 172-181, 2003.
18. RSA Labs: RSA Cryptography Standard: EMSAPSS-PKCS#1 v2.1, 2002.
19. A. Shamir, Identity based cryptosystems and signature schemes, Advances in Cryptology-Crypto'84, LNCS 196, Springer-Verlag, pages 47-53.
20. R. Sakai, K. Ohgishi, M. Kasahara. Cryptosystems based on pairing, In 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, 2000.
21. Z. F. Zhang, Y. B. Zhou and D. G. Feng. Efficient and Optimistic Fair Exchange based on Standard RSA with Provable Security, IACR Cryptology ePrint Archive, Report 2004/351, 2004.

# On the Quest for Impartiality: Design and Analysis of a Fair Non-repudiation Protocol

J. Cederquist<sup>1</sup>, R. Corin<sup>1</sup>, and M. Torabi Dashti<sup>2</sup>

<sup>1</sup> University of Twente

<sup>2</sup> CWI Amsterdam

**Abstract.** We design and analyze a simple optimistic fair non-repudiation protocol. Our protocol is considerably simpler and more efficient than current proposals, due mainly to the avoidance of using session labels. We model-check both safety and liveness properties. The safety properties are verified using a standard intruder, and the liveness properties using an intruder that respects the resilient communication channels assumption. Finally, to provide further confidence in the protocol, several vulnerabilities on weaker versions of our protocol are exposed.

## 1 Introduction

During the last decades the use of open networks for exchanging information has undergone an impressive growth. As a consequence, new security issues like *non-repudiation* and *fair exchange* have to be considered. Repudiation is the denial of a previously uttered statement. In the situation where agent  $A$  sends a message to agent  $B$ , non-repudiation guarantees that  $A$  cannot deny having sent the message and that  $B$  cannot deny having received it. One of the major difficulties in designing non-repudiation protocols is to achieve fairness, i.e. to avoid that one of the entities gets its evidence without the other one being able to get its evidence as well.

It has been shown that achieving fair exchange is impossible without a trusted third party (TTP) [18]. However, using a TTP in every exchange is inefficient. So, to avoid bottlenecks, Asokan et al. [2] introduced the optimistic approach to fair exchange, where the TTP is used *only* in the case of session recovery or abortion (which are assumed to be infrequent).

In comparison to other security issues like secrecy or authentication, fairness has not been studied intensively. Secrecy and authentication are safety properties for which the Dolev-Yao intruder is the most powerful intruder [7] (under certain assumptions, such as perfect cryptography). However, we also aim at verifying (session) termination, a liveness property that cannot be verified using the standard Dolev-Yao model. Therefore, we use a modified Dolev-Yao intruder that respects the resilient communication channels assumption (saying that messages sent over the network will eventually be delivered) [6].



In the literature, several fair non-repudiation protocols have been proposed, e.g. [14, 21, 13]. These protocols use *session labels* to identify session runs. A session label typically consist of a hash of all message components. However, using session labels does not only add computational cost, but also it may introduce vulnerabilities, as shown in [13].

In this paper we design an optimistic non-repudiation protocol which avoids using session labels altogether, and use a model checker to verify it. We refer the interested reader to [6] for detailed explanations regarding the adopted analysis technique and its comparison to other analysis approaches in the literature.

*Contributions.* Our contributions are threefold, as listed below.

- We propose a fair non-repudiation protocol which is simpler than existing ones. Existing fair non-repudiation protocols use labels to identify the session runs. Here we show that these labels can be avoided, allowing for a more efficient protocol. Our TTP distinguishes session runs by recognizing fresh keys, which the TTP receives in abort or resolve requests.
- We check a finite state model of our protocol, under the perfect cryptography assumption [8], using the technique of [6], briefly presented in Section 3. Our verification shows that an honest agent (that follows the protocol) will not be treated in an unfair way, even if the agent communicates with a dishonest agent that does not follow the protocol.
- To further validate the analysis method, we illustrate several vulnerabilities found by the model-checker when different fields are missing from our protocol. This also provides confidence in the full protocol and indicates that the fields are indeed needed.

The rest of the paper is organized as follows. In the next section we describe our fair non-repudiation protocol. The intruder model and the formal analysis are described in Section 3. In Section 4 we present the results of our formal analysis. We conclude with some related work and final remarks in Section 5.

## 2 A Fair Non-repudiation Protocol

In this section we describe our fair non-repudiation protocol. We first describe the underlying cryptographic assumptions and requirements on the trusted third party (TTP). Then we present our protocol, and finally we describe the evidences each party obtains and the fair exchange properties the protocol satisfies.

*Cryptographic assumptions.* In our analysis the cryptographic operations are assumed to be “ideal”, as in Dolev and Yao [8]: First, we assume to have a secure one way hash function  $h$ . Also, we have symmetric encryption of a message  $M$  with key  $K$ , denoted by  $\{M\}_K$ . In  $\{M\}_K$ ,  $M$  can only be extracted using  $K$ . We let  $\{K\}_{TTP}$  denote  $K$  encrypted asymmetrically using the trusted third party  $TTP$ 's public key. Finally,  $(X)_A$  denotes  $X$  signed by  $A$  (using  $A$ 's private key). The signature can be verified using  $A$ 's public key, and  $X$  can be extracted.

*TTP assumptions.* Our TTP is assumed to have a persistent database of aborted or resolved sessions, containing entries of the form  $\langle status : X Y W Z \rangle$ . In our protocol, *status* is either *aborted* or *resolved*;  $X$  and  $Y$  are agent identities,  $W$  is a cryptographic key and  $Z$  is the hash of a cipher text. A *query* to this database is denoted by a predicate  $status(X, Y, W, Z)$ , which holds when entry  $\langle status : X Y W Z \rangle$  exists in the TTP's database.

## 2.1 Protocol

The non-repudiation protocol that we present below allows an agent  $A$  to send a message  $M$  to agent  $B$  in a *fair* manner, meaning that  $A$  gets *evidence of receipt* (EOR) iff  $B$  receives  $M$  as well as *evidence of origin* (EOO). The EOR allows  $A$  to prove that  $B$  did indeed receive  $M$ , whilst the EOO allows  $B$  to prove that it was  $A$  who sent  $M$ . The protocol consists of three sub-protocols:

*Main protocol.* Agent  $A$  wants to send  $M$  to  $B$ , using *TTP* for session abort or resolution. Initially,  $A$  chooses a fresh key  $K$ . The main protocol is:

1.  $A \rightarrow B : \{M\}_K, EOO_M$  for  $EOO_M = (B, TTP, h(\{M\}_K), \{K, A\}_{TTP})_A$
2.  $B \rightarrow A : EOR_M$  for  $EOR_M = (EOO_M)_B$
3.  $A \rightarrow B : K$
4.  $B \rightarrow A : EOR_K$  for  $EOR_K = (A, h(\{M\}_K), K)_B$

First  $A$  sends  $\{M\}_K$ , along with  $EOO_M$ , which consists of  $B$  and *TTP*'s identities, a commitment to send  $M$  using  $K$  in the form of a hash  $h(\{M\}_K)$ , and  $K$  encrypted with the *TTP*'s public key (along with  $A$ 's identity) in case the session is later resolved. On receipt,  $B$  stores  $\{M\}_K$ , checks the signature of  $EOO_M$  to ensure that the message is genuinely coming from  $A$ , and extracts the values for performing more tests: Firstly, it checks that the leftmost value of  $EOO_M$  is  $B$ 's identity; Secondly, that *TTP* is a valid TTP for  $B$ , whom  $B$  trusts for recovering a session; Thirdly,  $B$  checks that the included hash commitment is indeed the hash of  $\{M\}_K$ . When all this is verified,  $B$  signs  $EOO_M$  with his private key to obtain  $EOR_M$  and sends it to  $A$ . When  $A$  gets this message, it checks whether the signature is that of  $B$ . If this is the case,  $A$  sends  $K$  to  $B$ . Then  $B$  sends  $EOR_K$ , signing  $K$  along with  $A$ 's identity and  $h(\{M\}_K)$ . Note that  $B$  does not need to check whether the key in message 3 decrypts  $\{M\}_K$ , since  $(A, h(\{M\}_K), K')_B$  would not be a valid evidence of receipt of  $M$  for  $A$ .

*Abort protocol.* If  $A$  does not receive a valid  $EOR_M$  from  $B$ , at step 2 in the main protocol, then  $A$  can invoke the *abort* protocol, for canceling the exchange:

1.  $A \rightarrow TTP : (abort, h(\{M\}_K), B, \{K, A\}_{TTP})_A$
2.  $TTP \rightarrow A : \begin{cases} E_{TTP} \text{ for } E_{TTP} = (A, B, K, h(\{M\}_K))_{TTP}, \\ \quad \text{if } resolved(A, B, K, h(\{M\}_K)) \\ AB_{TTP} \text{ for } AB_{TTP} = (A, B, h(\{M\}_K), \{K, A\}_{TTP})_{TTP}, \\ \quad \text{otherwise} \end{cases}$

First  $A$  sends to  $TTP$  an *abort request* message consisting of an *abort* flag, the commitment  $h(\{M\}_K)$ ,  $B$ 's identity and  $\{K, A\}_{TTP}$ . On receipt,  $TTP$  checks  $A$ 's signature, and checks that it can decrypt the message  $\{K, A\}_{TTP}$ . If the decryption succeeds,  $TTP$  checks that the included identity  $A$  next to the key  $K$  matches the signature of the whole abort request message. Next,  $TTP$  queries its database with  $resolved(A, B, K, h(\{M\}_K))$ . If the query holds, it means that this session has been resolved earlier. The answer from  $TTP$  to  $A$  is then  $E_{TTP}$ , including the key  $K$  signed by the private key of  $TTP$ . In the case that the query fails,  $TTP$  declares that the session is aborted and stores the entry  $\langle aborted : A B K h(\{M\}_K) \rangle$  in its database. The answer  $AB_{TTP}$  signed by the  $TTP$  is returned to  $A$ , as an acknowledgment of the successful abortion. Note that this message does not include  $K$  in the clear.

*Resolve protocol.* If  $B$  does not get  $K$  or  $A$  does not get  $EOR_K$ , then both parties may *resolve* the protocol by consulting  $TTP$ :

1.  $P \rightarrow TTP : ((B, TTP, h(\{M\}_K), \{K, A\}_{TTP})_A)_B$
2.  $TTP \rightarrow P : \begin{cases} AB_{TTP}, & \text{if } aborted(A, B, K, h(\{M\}_K)) \\ E_{TTP}, & \text{otherwise} \end{cases}$

Here  $P$  is the party that is resolving the session (i.e.  $A$  or  $B$ ). First  $P$  sends  $EOR_M$ , as a *resolve request* message. On receipt,  $TTP$  checks the validity of the signatures, and the successful decryption and matching of  $\{K, A\}_{TTP}$ . Then,  $TTP$  queries its database for  $aborted(A, B, K, h(\{M\}_K))$  to find out whether the session has been previously aborted. If the session has not been aborted, the resolve request is accepted and  $TTP$  stores  $\langle resolved : A B K h(\{M\}_K) \rangle$  in its database, and answers with  $E_{TTP}$  evidence containing key  $K$ , which is signed with  $TTP$ 's private key. If the session is already aborted,  $TTP$  answers with  $AB_{TTP}$ , a message representing the session abortion.

## 2.2 Evidences and Dispute Resolution

In case of a dispute, the parties present evidences to an external judge. In our protocol, the evidence of receipt EOR for  $A$  is  $EOR_M$  and  $\{M\}_K$ , plus either  $EOR_K$  or  $E_{TTP}$ . The evidence of origin EOO for  $B$  is  $EOO_M$ ,  $\{M\}_K$  and  $K$ .

*Dispute resolution.* Suppose  $B$  claims that it did not receive  $M$  from  $A$ , when  $A$  possesses EOR. Then  $A$  presents  $EOR_M$ ,  $\{M\}_K$  and either  $EOR_K$  or  $E_{TTP}$  to the judge. The messages  $EOR_M$  and  $\{M\}_K$  provide proof that  $B$  committed in the session to receive  $M$ , while  $EOR_K$  or  $E_{TTP}$  represent that either  $B$  received  $K$ , or he can receive it from  $TTP$ , respectively.

Suppose  $A$  claims that it did not send  $M$  to  $B$ , when  $B$  possesses EOO. Then  $B$  presents  $EOO_M$ ,  $\{M\}_K$  and  $K$  to the judge, who can check that  $A$  had indeed committed to communicate  $M$  to  $B$ . Since  $K$  was freshly created by  $A$ ,  $B$  could only have received it from  $A$  directly or from  $TTP$ , who checked that  $A$  provided the correct  $K$  in  $EOO_M$ .

## 2.3 Fair Exchange Properties

We aim at verifying *effectiveness*, *fairness* and *timeliness* (cf. requirements for fair exchange in [1]). These properties are illustrated in the case where  $A$  is the initiator and  $B$  the responder:

- Effectiveness says that if  $A$  and  $B$  behave according to the protocol and  $A$  does not abort, then the protocol session will reach a state where  $B$  has received the message  $M$  and EOO, and  $A$  has received EOR, and both  $A$  and  $B$  *terminate*, i.e. have no further pending operations to perform in that protocol session.
- Fairness expresses that when the protocol session has terminated then  $B$  has received  $M$  and EOO if and only if  $A$  has received EOR.
- Timeliness means that protocol sessions terminate for all honest parties. In other words, after an honest agent  $X$  has initiated a protocol session with some  $Y$ , then  $X$  will reach its termination<sup>1</sup>. Moreover, timeliness also specifies that *after* this termination the *degree of fairness* does not decrease for  $X$ : if  $X$  did not get his evidence before termination then it cannot be that  $Y$  gets her evidence without  $X$  also getting his.

Effectiveness is a functional sanity check, and may thus be verified in a system without intruder. For the other two properties, we can first verify termination and then check fairness and timeliness assuming that the protocol sessions terminate. This has the benefit of reducing the two properties to safety properties. Thus, termination is the only *liveness* property that needs to be checked.

## 3 Formal Analysis

We now implement the necessary machinery to formally analyze whether the protocol proposed in Section 2.1 meets the properties described in Section 2.3.

### 3.1 Communication Model

We consider two different communication models. The first model is used for verifying effectiveness. In this model there is no intruder (all agents are honest): A set of agents communicate over a network, performing send and receive actions. These actions are synchronized, meaning that an agent  $A$  can only send a message  $m$  to  $B$  (denoted by  $send(A, m, B)$ ), if  $B$  at the same time receives it from  $A$  (denoted by  $recv(A, m, B)$ ). The synchronization between  $send(A, m, B)$  and  $recv(A, m, B)$  actions is denoted by  $com(A, m, B)$ .

We use a second model to verify all the remaining properties. In this model there is an intruder  $I$  with complete control over the network. When an agent  $A$  sends a message  $m$  with the intention that it should be received by  $B$ , it is in fact the intruder that receives it, and it is also only from the intruder that  $B$  may receive  $m$ . Also in this model send and receive actions are synchronized.

<sup>1</sup> Here termination refers to that particular agents' session. An agent  $X$  may continue executing subsequent sessions after one session is finished.

### 3.2 The $\mu$ CRL Specification Language and Toolset

We briefly describe the symbols used in the  $\mu$ CRL code of the intruders below. For a complete description of the syntax and semantics of  $\mu$ CRL we refer to [12]. The symbols ‘.’ and ‘+’ are used for the sequential and alternative composition (“choice”) operator, respectively. The operator  $\sum_{d \in D} P(d)$  behaves like  $P(d_1) + P(d_2) + \dots$ . The process expression  $p \triangleleft b \triangleright q$ , where  $b$  is a Boolean term and  $p$  and  $q$  are processes, behaves like  $p$  if  $b$  is true, and like  $q$  if  $b$  is false. Finally,  $\tau$  represents an internal action, and the constant  $\delta$  expresses that, from then on, no action can be performed.

The formalization of the protocol described in Section 2 is carried out in  $\mu$ CRL [12]. The  $\mu$ CRL toolset includes an automatic state space generator and symbolic state space reduction tools. The fair exchange properties are expressed in the regular alternation-free  $\mu$ -calculus [16]. The model checker EVALUATOR 3.0 [16] from the CADP tool set [9] is then used to verify these properties.

### 3.3 Intruder Models

We use two different intruder models. For safety properties the normal Dolev-Yao intruder [8] is used. As mentioned earlier, this intruder is not suitable for verification of liveness properties [17], so to verify termination we use the intruder suggested in [6]. This intruder is shown to be equivalent, w.r.t. termination, to the Dolev-Yao intruder that respects the resilient communication channels assumption (RCC, messages sent over the network will eventually be delivered) [6], which is enough for our purposes. The Dolev-Yao intruder stores all received messages in a set  $X$ , representing its knowledge. The intruder uses  $X$  for synthesizing new messages (*synth* in the code below), using the usual rules of message (de)composition (in particular, the intruder can decrypt and sign messages only if it knows the corresponding key). The intruder can also block communications. Below we illustrate a specification of an intruder  $DY_B$ , in this case played by dishonest agent  $B$ . The intruder  $DY_B$  can perform a special evidence action  $evidence_B(k, m)$ . This action is parameterized by a key  $k$  and a message  $m$ , meaning that the gathered evidence regards message  $m$  and was provided in the session using key  $k$ . We allow  $DY_B$  to perform the action  $evidence_B(k, m)$  only when it can synthesize  $EOO(k, m)$ . In general, the particular data that constitutes an evidence is protocol specific, denoted below by  $EOO(k, m)$ .

$$\begin{aligned}
 DY_B(X) = & \sum_{\substack{p \in Agent \\ m \in Message}} recv(p, m, B).DY_B(X \cup \{m\}) + \\
 & \sum_{\substack{p \in Agent \\ synth(m, X)}} send(B, m, p).DY_B(X) + \\
 & \sum_{\substack{k \in Key \\ m \in msg}} evidence_B(k, m).DY_B(X) \triangleleft synth(EOO(k, m)) \triangleright \delta + \\
 & \tau.\delta
 \end{aligned}$$

According to the operational semantics that underlies  $\mu\text{CRL}$ , a process  $p + \delta$  behaves like  $p$ . So to express that the intruder shall be able to stop all communications at its own will, we let it perform an internal action  $\tau$  before deadlock  $\delta$ .

The intruder  $I_B$  for verifying termination maintains, besides  $X$ , a set  $Y$  for messages that have been received but not yet sent (cf. RCC). To distinguish the send actions that the intruder eventually has to perform (according to RCC) from the ones that it can perform (but does not have to), the send actions are tagged with  $X$  and  $Y$ , respectively. The synchronizations between send and receive actions are denoted  $com$ ,  $com_X$  and  $com_Y$  referring to the synchronizations between  $send$  and  $recv$ ,  $send_X$  and  $recv$ , and  $send_Y$  and  $recv$ , respectively.

$$\begin{aligned}
 I_B(X, Y) = & \sum_{\substack{p \in \text{Agent} \\ m \in \text{Message}}} recv(p, m, B).I_B(X \cup \{m\}, Y \cup \{m\}) + \\
 & \sum_{\substack{p \in \text{Agent} \\ m \notin Y \\ \text{synth}(m, X)}} send_X(B, m, p).I_B(X, Y) + \\
 & \sum_{\substack{p \in \text{Agent} \\ m \in Y}} send_Y(B, m, p).I_B(X, Y \setminus \{m\})
 \end{aligned}$$

Note that when we split the fairness and timeliness properties into termination and two safety properties, as described in Section 2.3, we also verify these properties using respectively the two intruders above. This can be done since the intruder  $I_B$  is equivalent to the Dolev-Yao intruder that respects the communication channels assumption [6].

### 3.4 Regular Alternation-Free $\mu$ -Calculus

The regular alternation-free  $\mu$ -calculus is used here to formulate properties of (states in) labeled transition systems. It is a fragment of  $\mu$ -calculus that can be efficiently checked. Here we briefly describe what is needed for expressing the fair exchange properties of the protocol we investigate. For a complete description of the syntax and semantics we refer to [16]. The regular alternation-free  $\mu$ -calculus is built up from three types of formulas: *action formulas*, *regular formulas* and *state formulas*. We use ‘.’, ‘ $\vee$ ’, ‘ $\neg$ ’ and ‘\*’ for concatenation, choice, complement and transitive-reflexive closure, respectively, of regular formulas. The symbols  $F$  and  $T$  are used in both action formulas and state formulas. In action formulas they represent *no action* and *any action*, respectively. The meaning of  $F$  and  $T$  in state formulas are the empty set and the entire state space, respectively. The operators  $\langle \dots \rangle$  and  $[\dots]$  have their usual meaning ( $\diamond$  and  $\square$  in modal logics). The CADP toolset also allows wildcards ‘\*’ in action parameters.

### 3.5 The Fair Exchange Properties

Here we formalize the fair exchange properties that we verify. To enhance readability, the protocol implementations are extended with certain *abstract* actions

that do not affect the behavior of the agents. An agent  $P$  performs the actions  $init_P(k, m)$  when it engages in a protocol session,  $terminate_P(k, m)$  when the session is over from  $P$ 's point of view, and  $evidence_P(k, m)$  when it receives a valid evidence, for the key  $k$  and item  $m$ . The  $TTP$  performs  $abort(k, m)$  when a session is successfully aborted, for the key  $k$  and item  $m$ .

First, we check that the protocol is effective. Note that this is verified in the model without intruder. Whenever agent  $P$  has started execution, then  $P$ 's termination is inevitable:

$$[T^*.init_P(k, m)] \mu Z. (\langle T \rangle T \wedge [\neg terminate_P(k, m)] Z) \quad (1)$$

Also, if there is no abort,  $P$  receives its evidence before termination:

$$[(\neg(abort(k, m) \vee evidence_P(k, m)))^*.terminate_P(k, m)] F \quad (2)$$

Now we turn to the fairness and timeliness properties, to be verified in the model with intruder. We assume that the intruder plays the role of  $Q$ . The properties below are thus defined to describe “fairness for  $P$ ”. The corresponding properties for  $Q$  are defined in a similar way. The properties fairness and timeliness are verified, as described above, by verifying termination separately, using the intruder described in Section 3.3

$$[T^*.init_P(k, m).(\neg terminate_P(k, m))^*] \langle (\neg com_X(*, *, *))^*.terminate_P(k, m) \rangle T, \quad (3)$$

i.e. whenever  $init_P(k, m)$  has happened, but not yet  $terminate_P(k, m)$ , there is a path to  $terminate_P(k, m)$  that does not contain  $com_X$  actions. This means that, whenever  $init_P(k, m)$  has happened, but not yet  $terminate_P(k, m)$ , and assuming RCC,  $terminate_P(k, m)$  will happen.

The remaining properties concern safety so we use the normal Dolev-Yao intruder. Fairness (for  $P$ ) means that if  $Q$  gets its evidence, then so shall  $P$ :

$$[(\neg evidence_Q(k, m))^*.evidence_P(k, m).(\neg evidence_Q(k, m))^*.terminate_P(k, m)] F \quad (4)$$

This property says that  $P$  does not terminate in an unfair state for  $P$ . But since  $P$  will eventually terminate (property 3),  $P$  will indeed terminate in a fair state.

Finally, timeliness for  $P$  says that after  $P$ 's termination, if  $P$  has not got his evidence,  $Q$  cannot get her evidence unless  $P$  initiates a new session with same key and item:

$$[(\neg evidence_P(k, m))^*.terminate_P(k, m).(\neg init_P(k, m))^*.evidence_Q(k, m)] F \quad (5)$$

In the case when  $P$  does initiate a new session with same key and item,  $P$  will get his evidence if  $Q$  gets hers (according to the properties 3 and 4).

## 4 Results

In this section we describe the results obtained from the formal analysis described in Section 3 performed on our protocol proposed in Section 2.

*Honest scenario  $S_0$ :  $A$  and  $B$  are honest.* We first encode a scenario in which both  $A$  and  $B$  are honest, along with the TTP.  $A$  exchanges items with  $B$  using fresh keys. To model timeouts, we use nondeterministic choices between communication actions. For instance, either  $A$  receives an answer timely from  $B$  in Message 2, in the main protocol, or it initiates the abort sub-protocol. Correspondingly,  $B$  has a choice between a send action and a  $\tau$  action, which models an asynchronous communication in which the message is ignored by  $A$ . This scenario was model-checked and showed to be deadlock-free and effective.

**Result 1.** *The protocol in Section 2.1 is effective for scenario  $S_0$ , satisfying the properties (1) and (2) in Section 3.5.*

*Dishonest scenario  $S_1$ :  $A$  dishonest and  $B$  honest.* When  $A$  is dishonest and  $B$  is honest, we execute  $B$  along with the intruder, who takes the identity of  $A$ . We first verify the safety properties (4) and (5) using the standard intruder (the first intruder in Section 3.3). Then we model check whether  $A$  can unfairly generate  $B$ 's evidence, and verify that this is impossible, thus rendering the protocol secure.

**Result 2.** *The protocol in Section 2.1 respects fair exchange and timeliness (properties (4) and (5) in Section 3.5, with respect to  $A$ ) for scenario  $S_1$ .*

Second, we force the intruder to respect RCC (by using the second intruder in Section 3.3). This scenario, called  $S'_1$ , is used to verify termination:

**Result 3.** *The protocol in Section 2.1 respects termination (property (3) in Section 3.5, with respect to  $A$ ) for scenario  $S'_1$ .*

*Dishonest scenario  $S_2$ :  $A$  honest and  $B$  dishonest.* In the opposite case, in which  $A$  is honest and  $B$  is dishonest, we obtain similar results to the above statements.

### 4.1 Further Experiments

We now illustrate vulnerabilities found by analyzing *modified versions* of the protocol presented in Section 2. The protocol is modified in such a way that certain assumptions are removed or different message components are excluded. The encountered vulnerabilities expose the need for the particular assumptions or excluded message components.

*Reuse of keys.* Suppose that  $A$  reuses a key  $K$  in a subsequent session. Then our analysis reports that for dishonest scenario  $S_2$ ,  $A$  may be attacked by  $B$ . The attack is reproduced in standard notation below:



- a1.  $A \rightarrow B : \{M\}_K, EOO_M$  for  $EEO_M = (B, TTP, h(\{M\}_K), \{K, A\}_{TTP})_A$
- $\vdots$
- b1.  $A \rightarrow B : \{M'\}_K, EEO_{M'}$  for  $EEO_{M'} = (B, TTP, h(\{M'\}_K), \{K, A\}_{TTP})_A$

First  $A$  sends the message a1, initiating a session. Then the session runs normally. When  $A$  later starts another session by sending message b1,  $B$  can immediately obtain  $M'$  and thus obtain the evidence EOO, before  $A$  can obtain its corresponding evidence. The vulnerability above was found in a scenario where  $A$  is honest, and uses two items and one key, and  $B$  is dishonest. This violation of property (4) shows that  $A$  needs to use fresh keys for each new session.

*Missing hash in  $EEO_M$ .* Consider  $EEO_M$ , the second component of the main protocol in Section 2.1. If we exclude the hash  $h(\{M\}_K)$ , obtaining a new  $EEO'_M = (B, TTP, \{K, A\}_{TTP})_A$ , the following vulnerability is found:

1.  $A \rightarrow B : \{M\}_{K'}, EEO'_M$  for  $EEO'_M = (B, TTP, \{K, A\}_{TTP})_A$
2.  $B \rightarrow A : EOR'_M$  for  $EOR'_M = (EEO'_M)_B$

Agent  $A$  starts a session with  $B$ , but uses a key  $K'$  to encrypt message  $M$  and embeds a *different* key  $K$  in  $EEO'_M$ . When  $B$  replies  $A$  can run the resolve protocol and obtain evidence EOR. However, when  $B$  wants to recover,  $TTP$  returns  $K$  which is not useful to decrypt  $\{M\}_{K'}$ , hence the evidences of  $A$  and  $B$  do not match. This vulnerability was found in a scenario where  $A$  is dishonest, and uses two keys, and  $B$  is honest. Again property (4) is violated which shows that including the hash in  $EEO_M$  is necessary for security of the protocol in Section 2.1.

*Missing  $A$ 's identity in  $EEO_M$ .* We now consider the case in which  $A$ 's identity is excluded from the component  $\{K, A\}_{TTP}$  in  $EEO_M$ . Suppose then that  $EEO'_M = (B, TTP, h(\{M\}_K), \{K\}_{TTP})_A$ . The following attack is found:

- a1.  $A \rightarrow B : \{M\}_K, EEO_M$  for  $EEO'_M = (B, TTP, h(\{M\}_K), \{K\}_{TTP})_A$
- b1.  $B \rightarrow C : \{M\}_K, EEO_M$  for  $EEO'_M = (C, TTP, h(\{M\}_K), \{K\}_{TTP})_B$
2.  $C \rightarrow B : EOR'_M$  for  $EOR'_M = (EEO'_M)_C$

When  $A$  starts a session with  $B$ ,  $B$  immediately starts another session with another agent  $C$ , reusing the information that  $A$  used. When  $C$  answers,  $B$  resolves and obtains  $K$  and hence the evidence. However  $A$  cannot obtain  $EEO_M$  since  $B$  never answers to  $A$ 's first message. When  $A$  is honest and  $B$  is dishonest,  $B$  can simply reuse its own identity and resolve to “itself” (we disallow the  $TTP$  to check this). Property (4) is thus violated, indicating that the identity of  $A$  is needed in  $\{K, A\}_{TTP}$ .

*Missing  $A$ 's identity in  $EOR_K$ .* If  $A$ 's identity is missing in  $EOR_K$  (so that  $EOR_K = (h(\{M\}_K), K)_B$ ), the following vulnerability is found:

- $a1. A \rightarrow B : \{M\}_K, EOO_M$  for  $EEO_M = (B, TTP, h(\{M\}_K), \{K, A\}_{TTP})_A$   
 $a2. B \rightarrow A : EOR_M$  for  $EOR_M = (EEO_M)_B$   
 $a3. A \rightarrow B : K$   
 $a4. B \rightarrow A : EOR_K$  for  $EOR_K = (h(\{M\}_K), K)_B$   
 $b1. C \rightarrow B : \{M\}_K, EOO_M$  for  $EEO_M = (B, TTP, h(\{M\}_K), \{K, C\}_{TTP})_C$   
 $b2. B \rightarrow C : EOR_M$  for  $EOR_M = (EEO_M)_B$

Here,  $A$  runs a normal session  $a$  with  $B$  which terminates.  $A$  is allied to another user  $C$ , who starts a replay of the session by  $A$ : we assume  $A$  hands over  $M$  and  $K$  to  $C$ . Now,  $B$  replies with  $EOR_M$ , at which point  $C$  aborts the session with  $B$ . Then  $B$  is unable to obtain evidence, but  $C$  has evidence since  $EOR_K$  does not mention  $A$  nor  $C$ , and thus it constitutes valid evidence EOR for  $C$  as well. This vulnerability appears in our analysis when we hand out information to a dishonest  $A$  about previous sessions giving some  $EOR_K$  to  $A$  (which may be from an old session of  $B$  with some other agent  $X$  which we assume is allied to  $A$ ). In such a scenario, property (4) is violated immediately when  $A$  runs the abort protocol after  $B$  answers its second message. Thus, we conclude that  $EOR_K$  needs to include  $A$ 's identity.

*Missing hash in  $EOR_K$ .* Finally we consider the case in which  $h(\{M\}_K)$  is missing in  $EOR_K$ , so  $EOR_K = (A, K)_B$ . The following attack is then possible:

- $a1. A \rightarrow B : \{M\}_K, EOO_M$  for  $EEO_M = (B, TTP, h(\{M\}_K), \{K, A\}_{TTP})_A$   
 $a2. B \rightarrow A : EOR_M$  for  $EOR_M = (EEO_M)_B$   
 $a3. A \rightarrow B : K$   
 $a4. B \rightarrow A : EOR_K$  for  $EOR_K = (A, K)_B$   
 $b1. A \rightarrow B : \{M'\}_K, EOO_M$  for  $EEO_M = (B, TTP, h(\{M'\}_K), \{K, A\}_{TTP})_A$   
 $b2. B \rightarrow A : EOR_M$  for  $EOR_M = (EEO_M)_B$

Similar to the previous case,  $A$  runs a normal session  $a$  with  $B$ . Then  $A$  starts another session, but now using a different message  $M'$ , reusing the same key  $K$ . After obtaining an answer from  $B$ ,  $A$  aborts the session. In this state  $A$  has valid evidence since the previous  $EOR_K$  is not bound to  $M$ , and thus it is valid also for an exchange between  $A$  and  $B$  with  $K$ . One could argue that  $B$  could also remember  $K$  and obtain  $M'$ . But  $B$  is not supposed to be stateful and save old keys,  $B$  just follows the protocol as is specified. In a scenario with  $A$  dishonest and  $B$  honest, property (4) is violated, exposing the mentioned vulnerability. This shows that  $EOR_K$  has to contain  $h(\{M\}_K)$ .

## 5 Conclusion and Related Work

We present a novel optimistic non-repudiation protocol, simpler than previous proposals. The simplicity is due to avoiding the usage of labels to identify sessions and assuming the usage of fresh keys per-session. We model-check the proposed protocol and verify the fair exchange properties using the technique in [6]. A full formalization can be found in an extended version of this document [5]. To

provide further confidence in our proposal we illustrate vulnerabilities when different fields are missing in the protocol.

*Related Work.* Several non-repudiation and fair exchange protocols have been previously proposed. Early tries on optimized exchange protocols [11, 10], i.e. those with only three message exchanges in honest protocol runs, have been found flawed [4, 20]. A recent optimized protocol suggested by Zhou [20], developed on previous ones, remarkably does not suffer from previously reported problems. But it has an elaborate dispute resolution phase requiring both participants to attend the court. We believe an evidence of receipt or origin must be self sufficient to settle a dispute, which requires the addition of a fourth message in our protocol.

More recently, Kremer and Markowitch [14] proposed a four-message protocol (KM) to achieve non-repudiation. Their protocol is analyzed by Kremer and Raskin [15] using a game-based technique. Quite similar to the KM protocol is Zhou-Gollman's protocol (ZG) [21]. Gürgens et al. [13] present several potential unfair situations that may happen in both the KM<sup>2</sup> and ZG protocols. These unfair situations arise from confusion in the *labels* used to identify the session runs. By carefully setting (complex) labels, Gürgens et al. propose a protocol for achieving fair exchange. The ZG protocol was also analyzed by Bella and Paulson [3] who used the theorem prover Isabelle to model the protocol by an inductive definition and to prove some desired properties. Another interesting approach to formal verification of fair exchange is the work by Shmatikov and Mitchell [19] who used the model checker Mur $\phi$  to analyze fair exchange and contract signing protocols, using a Dolev-Yao intruder.

*Acknowledgments.* We thank Ana Almeida Matos, Sandro Etalle, Wan Fokkink, Pieter Hartel, Steve Kremer and the anonymous reviewers for helpful comments.

## References

1. N. Asokan. *Fairness in electronic commerce*. PhD thesis, University of Waterloo, 1998.
2. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *4th ACM Conference on Computer and Communications Security*, pages 7–17. ACM Press, 1997.
3. G. Bella and L. C. Paulson. Mechanical proofs about a non-repudiation protocol. In R. J. Boulton and P. B. Jackson, editors, *TPHOLs 2001*, volume 2152 of *LNCS*, pages 91–104. Springer-Verlag, September 2001.
4. C. Boyd and P. Kearney. Exploring fair exchange protocols using specification animation. In *Information Security Workshop (ISW)*, volume 1975 of *LNCS*, pages 209–223. Springer-Verlag, 2000.
5. J. Cederquist, R. Corin, and M. Torabi Dashti. On the quest for impartiality: Design and analysis of a fair non-repudiation protocol (extended version). Technical Report TR-CTIT-05-32, University of Twente, The Netherlands, 2005.

---

<sup>2</sup> The KM protocol was not originally designed to provide fair exchange but simply non-repudiation (private communication, 2004).

6. J. Cederquist and M. Torabi Dashti. An intruder model for verifying termination in security protocols. Technical Report TR-CTIT-05-29, University of Twente, Enschede, The Netherlands, 2005.
7. I. Cervesato. The Dolev-Yao Intruder is the Most Powerful Attacker. In J. Halpern, editor, *LICS'01*, Boston, MA, 16–19 June 2001. IEEE Computer Society Press.
8. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2):198–208, March 1983.
9. J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A protocol validation and verification toolbox. In R. Alur and T. A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification*, volume 1102 of *LNCS*, pages 437–440. Springer-Verlag, 1996.
10. J. Ferrer-Gomila, M. Payeras-Capella, and L. Huguet i Rotger. A realistic protocol for multi-party certified electronic mail. In *Proceedings of the 5th International Conference on Information Security*, pages 210–219, UK, 2002. Springer-Verlag.
11. J. L. Ferrer-Gomila and L. H. Rotger. An efficient asynchronous protocol for optimistic certified mail. In *International Workshop on Cryptographic Techniques and E-Commerce (Cryptec)*, 1999.
12. J. F. Groote and A. Ponse. The syntax and semantics of  $\mu$ CRL. In A. Ponse, C. Verhoef, and S. F. M. van Vlijmen, editors, *Algebra of Communicating Processes '94*, Workshops in Computing Series, pages 26–62. Springer-Verlag, 1995.
13. S. Gürgens, C. Rudolph, and H. Vogt. On the security of fair non-repudiation protocols. In *Information Security Conference (ISC)*, volume 2851 of *LNCS*, pages 193–207. Springer, 2003.
14. S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, November 2002.
15. S. Kremer and J. Raskin. A game-based verification of non-repudiation and fair exchange protocols. In K. Larsen and M. Nielsen, editors, *Proceedings of the 12th International Conference on Concurrency Theory*, volume 2154 of *LNCS*, pages 551–565. Springer-Verlag, 2001.
16. R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *Sci. Comput. Program.*, 46(3):255–281, 2003.
17. C. Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE Journal on Selected Areas in Communication*, 21(2):44–54, 2003.
18. H. Pagnia and F. C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University, 1999.
19. V. Shmatikov and J. C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science*, 283(2):419–450, 2002.
20. J. Zhou. On the security of a multi-party certified email protocol. In *Proc. ICICS'04*, volume 3269 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 2004.
21. J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 55–61, Oakland, CA, 1996. IEEE Computer Society Press.

# Generic, Optimistic, and Efficient Schemes for Fair Certified Email Delivery

Guilin Wang<sup>1</sup>, Feng Bao<sup>1</sup>, Kenji Imamoto<sup>2</sup>, and Kouichi Sakurai<sup>2</sup>

<sup>1</sup> Institute for Infocomm Research,  
21 Heng Mui Keng Terrace, Singapore 119613

{glwang, baofeng}@i2r.a-star.edu.sg

<sup>2</sup> Kyushu University, Fukuoka, Japan

imamoto@itslab.csce.kyushu-u.ac.jp

sakurai@csce.kyushu-u.ac.jp

**Abstract.** As a value-added service for standard email systems, a certified email scheme allows a sender to deliver a message to a receiver in a *fair* way in the sense that either the sender obtains a receipt from the receiver and the receiver accesses the content of the email simultaneously, or neither party gets the expected item. In this paper, we first point out some weaknesses in several existing schemes. Then, we present two generic optimistic certified email schemes with transparent TTP. Our schemes are not only fair, but also support timeliness in two flavors: one scheme supports weak timeliness but with stateless TTP, while the other guarantees (strong) timeliness though only supports weak stateless TTP. Technical discussion and comparison are provided to show that our schemes are both secure and efficient, compared with the-state-of-art in this field.

**Keywords:** certified email, non-repudiation, fair exchange, security protocol.

## 1 Introduction

Certified email schemes are designed to achieve fair-exchange of a message and a receipt between two potentially mistrusting parties over the Internet. We call such a scheme *fair*, if it is guaranteed that the message receiver can get the email content *if and only if* the message sender obtains an irrefutable receipt from the receiver. In other words, a dishonest party cannot obtain his/her expected item from a honest party in a cheating way such that the honest party is unable to get the corresponding item.

The undeniable receipt issued by the receiver serves as an evidence for non-repudiation of receipt (NRR). Namely, if the receiver denies having received the delivered message from the sender, the sender can provide publicly verifiable NRR evidence to an arbitrator to show that this is untrue. Some email schemes also provide evidences for non-repudiation of origin (NRO). Similarly, NRO evidence protects the receiver from the sender's dishonest denial that she has not deliver a particular message to the receiver, though this is the fact. We remark

that certified email schemes supporting NRO services are functionally equivalent to non-repudiation protocols.

Since direct fair-exchange between two parties is extremely inefficient on both aspects of computation and communication, realistic solutions for certified email delivery need a trusted third party (TTP). Actually, according to the different usage of the TPP, certified email schemes (and non-repudiation protocols) can be classified in two types: schemes with on-line TTPs [8,16] or schemes with off-line TTPs [17,1,18,2,10,3,9,12,14]. Generally speaking, it would not be too difficult to carry out the first type schemes, due to the TTP's participation in every protocol instance (though maybe not in each step). However, those schemes are still expensive and inefficient in practice, since the TTP must offer services with high level of availability and performance, and the TTP is likely to become the system bottleneck if numerous certified emails per day have to be exchanged via the same TTP. A more appealing and promising approach is to exploit the TTP in an off-line fashion. Actually, those schemes with off-line TTPs are called as *optimistic* [1], since the TTP is not invoked in the protocol execution at all, unless one of the two parties misbehaves or the communication channel is out of order. In fact, most of researches in this area have focused on those optimistic fair-exchange protocols.

Most of certified email schemes [17,18,10,3,9,12,14] are designed to satisfy some or all of the following properties:

- P1) **Optimism:** The TTP is involved only in abnormal cases, i.e., one party is trying to cheat or the communication channel fails to work.
- P2) **Generic Construction:** Each party can *independently* exploit *any* (secure) standard signature scheme to generate non-repudiation evidences.
- P3) **Transparent TTP:** The generated non-repudiation evidences are the same regardless of whether the TTP is involved in the protocol execution.
- P4) **Stateless TTP:** To settle potential disputes between users, the TTP is not required to keep a database for remembering and searching the state information for each protocol instance.
- P5) **High Performance:** To execute the protocol, both overheads of computation and communication could be reduced as much as possible.
- P6) **NRR Service:** The protocol generates NRR (non-repudiation of receipt) evidence for the sender, which proves that the receiver received a specific message from the sender, if the receiver indeed obtained this message.
- P7) **NRO Service:** The protocol generates NRO (non-repudiation of origin) evidence for the receiver, which proves that the sender delivered a specific message to the receiver, if the receiver indeed did this successfully.
- P8) **Fairness:** After the completion of a protocol run, either each party obtains the expected item or neither party does.
- P9) **Timeliness:** At any time during a protocol run, each party can *unilaterally* choose to terminate the protocol without losing fairness.
- P10) **Confidentiality:** Except the receiver and the sender, the content of the delivered message cannot be accessed by anybody else, including the TTP.

In the above list, the first five properties are very meaningful in real-world systems to reduce the running cost of the TTP. As for the security requirements, fairness and NRR evidence are essential requirements for all certified email schemes, while NRO evidence, timeliness and confidentiality are optional. Actually, some schemes may only meet *weak* timeliness, i.e., one party can only terminate the protocol after waiting for a fixed amount of time, not on any moment during the protocol execution.

However, the existing schemes do not satisfy part of the above ten properties. In some cases, fairness, the most important security requirement for certified email, cannot be achieved due to the existence of some subtle but reasonable attacks [12]. This paper has two main contributions. We first identify some weaknesses in the certified email schemes proposed in [9,14]. After that, we present two certified email schemes supporting as many as possible properties. Specifically, one of our schemes supports weak timeliness but with stateless TTP, while the other satisfies (strong) timeliness though only supports weak stateless TTP. Except the difference on those two properties, all other eight properties are guaranteed by both schemes simultaneously. Table 1 shows that our schemes are not only secure but also efficient, when we compared them with the-state-of-art in this field.

Note that the on-line schemes in [8,16] are not enumerated in Table 1, since we only focus on optimistic protocols. In addition, to evaluate the efficiency we just compare the costs of both communication and computation in *normal* case, i.e., in the exchange protocols, since the abort and recovery protocols are only run rarely in abnormal cases. “Messages” means the number of message flows need to be transferred between the sender and receiver in each exchange protocol, while “Operations” denotes the number of asymmetric operations need to be performed by the two parties. Fairness in most of those schemes is affected by different potential attacks, so we mark those schemes with “Yes\*” under the

**Table 1.** Comparison of Basic Features, Efficiency, and Security

	ZG [17]	ZDB [18]	GRV [12]	MK [13]	KM [10]	AN [3]	Micali [14]	IS [9]	New I	New II
P1. Optimism	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
P2. Generic Constr.	Yes	Yes	Yes	No	Yes	Yes*	Yes	Yes	Yes	Yes
P3. Trans. TTP	No	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes
P4. Stateless TTP	No	No	No	No	No	Yes	Yes	Yes	Yes	Weak
P5. Messages	4	4	4	4	4	4	3	3	3	3
P5. Operations	8	12	10	12	8	17	8	8	8	8
P6. NRR Service	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
P7. NRO Service	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
P8. Fairness	Yes*	Yes*	Yes	Yes*	Yes*	Yes	Yes*	Yes*	Yes	Yes
P9. Timeliness	Weak	Yes	Yes	Yes	Yes	No	Weak	No	Weak	Yes
P10. Confidentiality	No	No	No	No	No	No	Yes	Yes	Yes	Yes

column of “fairness”. Because those schemes are likely to be repaired by more or less modifications, though the security of revised schemes should be checked carefully again.

The rest of the paper is organized as follows. Section 2 introduces notations. In Section 3, we review and analyze the IS scheme [9]. After that, two new schemes are presented and then analyzed in Section 4 and Section 5, respectively. Finally, Section 6 concludes the paper.

## 2 Notations

In this paper, we use  $A$ ,  $B$ , and  $T$  to denote unique identifiers of a sender Alice, a receiver Bob, and a TTP, respectively.  $m$  is a message Alice wants to deliver to Bob.  $c = E_k(m)$  is the ciphertext of message  $m$  encrypted with a symmetric encryption algorithm  $E_k(\cdot)$ , where  $k$  is a session key selected by the sender Alice.  $D_k(\cdot)$  denotes the corresponding symmetric decryption algorithm, i.e., we have  $m = D_k(E_k(m))$ . In addition,  $H(\cdot)$  stands for a cryptographic hash function.  $E_X(\cdot)$  means party  $X$ 's asymmetric encryption algorithm, so that the resulting ciphertexts can only be decrypted by party  $X$  using its private key.  $S_X(\cdot)$  denotes party  $X$ 's secure signing algorithm, so that the resulting digital signatures can be verified by anybody using party  $X$ 's signature verification key, which is assumed to be publicly available.

## 3 The IS Scheme and Its Security

In [9], the authors actually proposed two certified email schemes, one with on-line TTP, while the other with off-line TTP. Here, we only review and analyze their scheme with off-line TTP. For simplicity, we call it the IS scheme.

### 3.1 Review of the IS Scheme

(1) **Exchange Protocol.** When Alice wants to deliver a message  $m$  to Bob, they jointly executed the following exchange protocol:

$$\begin{aligned} \text{(e1). } & A \longrightarrow B : A, T, c, EK, \text{sub}, S_A(B, c, EK) \\ \text{(e2). } & B \longrightarrow A : S_B(S_A(B, c, EK)) \\ \text{(e3). } & A \longrightarrow B : E_B(k) \end{aligned}$$

The whole exchange protocol is explained in detail as follows.

- 1). The sender Alice first generates a random session key  $k$ , then computes  $c = E_k(m)$ ,  $EK = E_T(A, B, E_B(k))$ , and her signature  $S_A(B, c, EK)$ . Then, Alice transmits  $(A, T, c, EK, \text{sub}, S_A(B, c, EK))$  to Bob, where **sub** is the subject of message  $m$  to encourage Bob receiving this encrypted email.
- 2). Upon receiving message flow (e1), Bob checks whether  $S_A(B, c, EK)$  is Alice's valid signature on message  $(B, c, EK)$ . If this is true and Bob would like to get this email from Alice, he returns back his signature  $S_B(S_A(B, c, EK))$  to the sender Alice. Otherwise, Bob could ignore message flow (e1).



- 3). If Bob's signature  $S_B(S_A(B, c, EK))$  is received timely and correctly, the sender Alice reveals  $E_B(K)$  to the receiver Bob.
- 4). When  $E_B(K)$  is received, the receiver Bob first derives the session key  $k$  by computing  $k = D_B(E_B(k))$ , and then obtains message  $m = D_k(c)$ .

(2) **Recovery Protocol.** If Bob honestly sent his receipt  $S_B(S_A(B, c, EK))$  to Alice but Alice does not reveal  $E_B(k)$  to him, Bob can initiate the following recovery protocol to get  $E_B(k)$  from the TTP directly.

$$\begin{aligned}
 \text{(r1). } & B \longrightarrow T : c, EK, S_B(S_A(B, c, EK)) \\
 \text{(r2). } & T \longrightarrow B : E_B(k) \\
 & T \longrightarrow A : S_B(S_A(B, c, EK))
 \end{aligned}$$

Upon receiving a recovery request (r1), the TTP first decrypts  $EK$  to get  $(A, B, E_B(k))$ , and then checks whether  $S_A(B, c, E_B(k))$  and  $S_B(S_A(B, c, EK))$  are valid signatures. If both of them are correct, the TTP forwards  $E_B(k)$  to Bob and  $S_B(S_A(B, c, EK))$  to Alice.

(3) **Dispute Resolution Policy.** The original authors do not provide explicit dispute resolution policy, though they briefly mentioned that the generated signatures can be used as non-repudiation evidences (page 333 in [9]).

### 3.2 Security of the IS Scheme

In this section, we point out some weaknesses in the IS scheme.

(1) **Protocol Specification.** In the IS scheme, the protocol specification is not clear enough in the following senses. Firstly, when the exchange protocol is finished, Bob also needs to check whether  $EK \equiv E_T(A, B, E_B(k))$ . Otherwise, what he received may be an invalid NRO evidence. Secondly, the authors do not provide explicit definitions of NRO and NRR evidences, and exact procedures how a judge (or verifier) can validate those evidences. Obviously, just providing  $S_B(S_A(B, c, EK))$  is not enough. Finally, there are no clear requirements on public encryption algorithms. That is, if those algorithms are randomized, how to deal with the random numbers utilized to produce  $E_B(k)$  and  $EK$ . Simply requiring Alice should reveal those numbers to Bob may be not sufficient.

(2) **An Attack.** In the real world, there may be many TTPs. One user, say Bob, is probably not aware the identities of all TTPs. In such a scenario, a dishonest sender Alice can mount the following attack to disturb or even successfully cheat the receiver Bob.

- 0). Before executing the protocol, Alice and Bob have agreed to use a specific  $T$  as their TTP for certified e-mail delivery. However, Alice dishonestly exploits  $T'$ 's public key to encrypt session key  $k$  by computing  $EK' = E_{T'}(A, B, E_B(k))$ , where  $T'$  is another TTP whose identity is unlikely known by Bob, for example in another city or country.
- 1). Then, Alice correctly prepares other values and sends the receiver Bob below message flow (e1'):  $(A, T, c, EK', \text{sub}, S_A(B, c, EK'))$ .

- 2). According to the IS scheme, message flow (e1') is correct, so honest Bob will return his signature  $S_B(S_A(B, c, EK'))$  to Alice.
- 3). After that, however, Alice refuses to reveal  $E_B(k)$  to Bob.
- 4). So Bob contacts the TTP  $T$  to get  $E_B(k)$  by providing  $c, EK', S_A(B, c, EK')$  and  $S_B(S_A(B, c, EK'))$ . But  $T$  will reject Bob's recovery request, since it cannot decrypt  $EK'$  correctly, due to the fact  $EK'$  is encrypted under the public key of  $T'$ .

The result of the above attack is that Bob would think the above protocol instance with Alice is unsuccessful (and then may delete related information sooner or later), but Alice has already gotten Bob's valid receipt, i.e.,  $(A, B, T', m, k, S_A(B, c, EK'), S_B(S_A(B, c, EK')))$ . By showing this receipt, any judge will believe that Bob has received message  $m$  from Alice. It seems unreasonable to assume that Bob will contact with all (possible) TTPs for a recovery request. To avoid this attack, the TTP's identity  $T$  could be added in  $S_A$  and  $S_B$ .

We remark that a similar attack applies the schemes in [14].

**(3) Efficiency.** The IS scheme's efficiency could be improved in two aspects.

(a) When Bob initiates the recovery protocol, he is required to send the TTP ciphertext  $c$  together with other values. However, if  $m$  is a file with large size, such as digital movies, the communication overhead and delay are increased. (b) In the IS scheme, the non-repudiation evidence for both origin and receipt are the same, i.e., two embedded signatures  $(S_A(B, c, EK), S_B(S_A(B, c, EK)))$ . So, to resolve a dispute the judge has to verify two signatures (and other values). In our new scheme, the judge only needs to validate one signature, and  $c$  will be replaced by  $H(c)$  in our recovery protocol.

## 4 The Proposed Schemes

In this section, we present two new certified email schemes, which are secure and more efficient than the IS scheme. As well as the IS scheme, our schemes are generic optimistic protocols with transparent TTPs, and support fairness, confidentiality, both NRR and NRO evidences. Moreover, our schemes meet one more property: timeliness, which is not supported in the IS scheme. More specifically, one proposed protocol satisfies weak timeliness but with stateless TTP, while the other guarantees (strong) timeliness but the TTP needs to store some state information. Naturally, both versions could find their advantages in different environments.

To support the confidentiality of delivered message, we use the same idea in the IS scheme, i.e.,  $E_B(k)$  is embedded in  $EK$ . However, almost all ingredients and sub-protocols are re-designed. Of course, in this process we are inspired by the ideas appear in the literature, especially those in [6,12,9,14]. We assume that  $E_B(\cdot)$  and  $E_T(\cdot)$  are CCA2-secure randomized encryption algorithms. To emphasize a random number  $r$  is used to encrypt a message  $M$ , we write  $C = E_T^r(M)$ . Furthermore, we assume that both  $E_B(\cdot)$  and  $E_T(\cdot)$  have the property of *randomness recoverability*. Namely, in the procedure of decryption,

the owner of the secret decryption key can recover not only the message  $M$  but also the random number  $r$  used to produce the ciphertext. For simplicity, we denote this fact by  $(M, r) = D_T(E_T^r(M))$ . Actually, the OAEP series [15] are typical examples for such cryptosystems. As usual, the communication channel between Alice and Bob is assumed to be *unreliable*, i.e., messages inserted into such a channel may be lost. However, the TTP is linked with Alice and Bob by *reliable* communication channels, i.e., messages inserted into such a channel will be delivered to the recipient after a finite delay. In real life, such a communication channel could be implemented via computer networks compensated by other communication means, such as fax, telephone, or express mail service etc.

In addition, we introduce a new symbol  $P$  to denote the unique identifier for our protocol.  $P$  explicitly specifies the protocol name, version, and the cryptographic algorithms employed. By the appearance of  $P$ , related signatures can only be interpreted in a special way according to our definitions. We believe this is more logical and reasonable in practice. The following notations are re-defined or newly introduced.

- $\ell = H(P, A, B, T, HC, HK, t)$ : A unique label to identify a protocol instance, where  $H(\cdot)$  is a secure hash function. Label  $\ell$  means that a message  $m$  is supposed to be delivered from the sender Alice to the receiver Bob (with/without the TTP's help), where  $m$  is determined by a ciphertext  $c$  and a symmetric key  $k$  such that  $m = D_k(c)$ ,  $HC = H(c)$  and  $HK = H(k)$ . Here,  $t$  denotes a deadline with the meaning that after the expiration of  $t$ , the TTP does not accept a resolution request related to this  $t$  anymore. Label  $\ell$  is used to identify a specific protocol instance, and link all messages generated in this instance.
- $EK = E_T^{r_2}(\ell, EB)$ : Encrypted session key that includes label  $\ell$  and  $EB = E_B^{r_1}(k)$ , where both  $r_1$  and  $r_2$  are random numbers. Compared with the IS scheme, both  $T$  and  $P$  are embedded in  $EK$  via label  $\ell$ .
- $S_A = S_A(P, \ell, EK)$ , and  $S_B = S_B(P, \ell, EK)$ : In contrast to the IS scheme, three changes are made here. (1) Non-repudiation evidences are defined as signatures on  $(P, \ell, EK)$ , not on  $(B, c, EK)$ . However, note that since label  $\ell$  is determined by  $(P, A, B, T, H(c), H(k), t)$ , so we have implicitly embedded the TTP's identifier  $T$  (and other information) in  $S_A$  and  $S_B$ . (2) Bob signs on message  $(P, \ell, EK)$  directly, instead of signing on Alice's signature  $S_A$ . (3) To generate and verify label  $\ell$ , the whole ciphertext  $c$  is not needed. So, in our recovery protocol, only  $H(c)$  instead of  $c$  is submitted to the TTP. By doing so, the communication overhead between Bob and the TTP is further reduced.

#### 4.1 Certified Email Scheme with Weak Timeliness

This new certified scheme consists of a exchange protocol, a recovery protocol, and a dispute resolution policy. Due to the absence of abort protocol, the TTP is not supposed to store any information related to a specific protocol instance. In addition, weak timeliness is achieved by the introduction of deadline  $t$ , which

could be negotiated by the two parties involved before the protocol executing. For example, let  $t$  be 24 hours after the beginning time of protocol execution.

**Exchange Protocol.** To send a message  $m$ , the sender Alice and receiver Bob execute the following exchange protocol collectively.

- (e1).  $A \longrightarrow B : P, A, B, T, c, HK, EK, t, S_A(P, \ell, EK)$
- (e2).  $B \longrightarrow A : S_B(P, \ell, EK)$
- (e3).  $A \longrightarrow B : EB, r_2$

That is, Alice first selects a session key  $k$  and two random numbers  $(r_1, r_2)$ , and then computes the following values:  $c = E_k(m)$ ,  $HC = H(c)$ ,  $HK = H(k)$ ,  $\ell = H(P, A, B, T, HC, HK, t)$ ,  $EB = E_B^{r_1}(k)$ ,  $EK = E_T^{r_2}(\ell, EB)$ , and  $S_A(P, \ell, EK)$ . After that, message flow (e1) is delivered to Bob. When Bob receives message flow (e1), he first recovers label  $\ell$  and checks whether  $S_A(P, \ell, EK)$  is indeed Alice's signature on  $(P, \ell, EK)$ . If this true, Bob further evaluates whether the included deadline  $t$  is sufficient for him to apply the TTP's help. If yes, Bob can return his signature  $S_B(P, \ell, EK)$  to the sender Alice as message flow (e2). Otherwise, if any of the above checks fails or he does not want to receive this message from Alice, Bob can simply reject this email without any liability. If Bob's valid signature  $S_B(P, \ell, EK)$  is received, Alice reveals  $(EB, r_2)$  to Bob. Finally, Bob checks whether  $EK \equiv E_T^{r_2}(\ell, EB)$ . If this true, Bob first decrypts  $(k, r_1)$  from  $EB$ , and then obtains the message  $m$  by computing  $m = D_k(c)$ . However, if Bob does not receive correct pair  $(EB, r_2)$  from Alice timely, he can execute the recovery protocol with the TTP (see below).

**Recovery Protocol I.** Whenever before the deadline  $t$ , Bob could initiate the following recovery protocol to get  $(EB, r_2)$  from the TTP directly.

- (r1).  $B \longrightarrow T : P, A, B, T, HC, HK, EK, t, S_A(P, \ell, EK), S_B(P, \ell, EK)$
- (r2).  $T \longrightarrow B : \ell, EB, r_2$   
 $T \longrightarrow A : \ell, S_B(P, \ell, EK)$

In detail, Bob first sends all related values to the TTP. Then, the TTP recovers label  $\ell$ , and checks the validity of deadline  $t$ ,  $S_A(P, \ell, EK)$ , and  $S_B(P, \ell, EK)$ . If everything is ok, it decrypts  $EK$  with its secret key. If the result is the expected pair  $(\ell, EB)$  with a random number  $r_2$ , the TTP forwards  $(\ell, EB, r_2)$  and  $(\ell, S_B(P, \ell, EK))$  to Bob and Alice, respectively. However, if  $EK$  cannot be decrypted successfully, the TTP informs Bob that his recovery request is invalid.

*Remark 1.* Note that as we mentioned before, our recovery protocol has the following two features. First, in the theory the TTP is not required to store any information about a specific recovery request. The TTP's work is just to check the validity of a request, and then gives the corresponding answer. It does not need to remember anything except its decryption secret key. This is, this certified email scheme supports stateless TTP, though it only satisfies weak timeliness. Second, the overhead of communication between Bob and the TTP is independent of the size of message  $m$ , since only hash value  $H(c)$ , instead of  $c$ , is delivered to the TTP in the recovery protocol.

*Remark 2.* In the above scheme, Alice may need to wait Bob's signature  $S_B$  until the expiration of deadline  $t$ . That is, this certified email scheme only supports weak timeliness, since Alice cannot terminate a protocol run at *any* time. However, once Bob successfully executed the recovery protocol (before deadline  $t$ ), Alice will receive the receipt from the TTP correctly. On the other hand, if Bob does not successfully apply recovery before the expiration of deadline  $t$ , this protocol run is deemed to be cancelled. In this situation, the result is still fair but unsuccessful, since neither Alice nor Bob can get their expected items.

### Dispute Resolution Policy I.

- **Non-Repudiation of Origin.** To show that Alice indeed delivered message  $m$  to himself, Bob can provide  $(P, A, B, T, m, k, r_1, r_2, t, S_A)$  to a judge. Then, the judge performs as follows:
  - 1) Compute  $c = E_k(m)$ ,  $EB = E_B^{r_1}(k)$ ,  $\ell = H(P, A, B, T, H(c), H(k), t)$ , and  $EK = E_T^{r_2}(\ell, EB)$ .
  - 2) Check whether  $S_A$  is Alice's valid signature on message  $(P, \ell, EK)$ . If yes, accept Bob's claim. Otherwise, reject Bob's claim.
- **Non-Repudiation of Receipt.** Similarly, to show that Bob has already received message  $m$  from herself, Alice can provide  $(P, A, B, T, m, k, r_1, r_2, t, S_B)$  to a judge. Then, the judge performs as follows:
  - 1) Compute  $c = E_k(m)$ ,  $EB = E_B^{r_1}(k)$ ,  $\ell = H(P, A, B, T, H(c), H(k), t)$ , and  $EK = E_T^{r_2}(\ell, EB)$ .
  - 2) Check whether  $S_B$  is Bob's valid signature on message  $(P, \ell, EK)$ . If yes, accept Alice's claim. Otherwise, reject Alice's claim.

## 4.2 Certified Email Scheme with (Strong) Timeliness

In this version, the exchange protocol is the same as in the previous version. To support (strong) timeliness, we need to add an abort protocol, and modified the recovery protocol so that those two sub-protocols could work consistently.

**Abort Protocol II.** If Alice already delivered message flow (e1) to Bob but does not receive the expected  $S_B(P, \ell, EK)$  correctly or timely, she can initiate the following abort protocol to cancel the protocol instance at *any* time before deadline  $t$ .

- (a1).  $A \longrightarrow T : P, A, B, T, HC, HK, EK, t, E_T^{r_3}(S_A(P, \ell, \text{abort}))$   
 if request is invalid then stop  
 if (state=aborted) then retrieve  $\ell, S_A(P, \ell, \text{abort})$   
 $T \longrightarrow A : \ell, \text{confirmation}$   
 if (state=recovered) then retrieve  $\ell, S_B(P, \ell, EK)$   
 $T \longrightarrow A : \ell, S_B(P, \ell, EK)$   
 else set (state=aborted) and store  $\ell, S_A(P, \ell, \text{abort})$
- (a2).  $T \longrightarrow A : \ell, \text{confirmation}$   
 $T \longrightarrow B : \ell, S_A(P, \ell, \text{abort})$

To do so, Alice first sends message flow (a1) to the TTP, where  $S_A(P, \ell, \text{abort})$  serves as an *abort token* and is encrypted under the TTP's public key. When the TTP received such an abort request, it first recovers label  $\ell$ , decrypts the last item, and then checks the validity of deadline  $t$  and signature  $S_A(P, \ell, \text{abort})$ . If any of those checks fails, the TTP rejects Alice's request. Otherwise, it further checks whether label  $\ell$  is already stored in its database. If yes, this means this protocol run identified by label  $\ell$  has been aborted or recovered successfully, so the TTP retrieves corresponding items and forwards them to Alice. Otherwise, it sets state variable as **aborted**, stores  $(\ell, S_A(P, \ell, \text{abort}))$  into its database, forwards  $(\ell, S_A(P, \ell, \text{abort}))$  to Bob, and gives a confirmation to Alice. Here, confirmation can be defined as the TTP's signature on  $(P, \ell, S_A(P, \ell, \text{abort}))$ .

**Recovery Protocol II.** Similarly, if Bob already sent  $S_B(P, \ell, EK)$  to Alice but does not get correct pair  $(EB, r_2)$  from Alice in a reasonable period before deadline  $t$ , he can get this pair directly from the TTP by executing the following recovery protocol.

- (r1).  $B \longrightarrow T : P, A, B, T, HC, HK, EK, t, S_A(P, \ell, EK), S_B(P, \ell, EK)$   
 if request is invalid then stop  
 if (state=aborted) then retrieve  $\ell, S_A(P, \ell, \text{abort})$   
 $T \longrightarrow B : \ell, S_A(P, \ell, \text{abort})$   
 if (state=recovered) then retrieve  $\ell, EB, r_2$   
 $T \longrightarrow B : \ell, EB, r_2$   
 else set (state=recovered) and store  $\ell, S_B(P, \ell, EK), EB, r_2$
- (r2).  $T \longrightarrow A : \ell, S_B(P, \ell, EK)$   
 $T \longrightarrow B : \ell, EB, r_2$

That is, Bob initially sends message flow (r1) to the TTP. Upon receiving such a recovery request, the TTP first recovers label  $\ell$ , checks the deadline, verifies the signatures, and decrypts the ciphertext  $EK$ . If any of the above operations is unsuccessful, Bob's request will be rejected. Otherwise, the TTP further checks whether the protocol instance identified by label  $\ell$  has been aborted or recovered successfully by searching its database. If yes, it retrieves corresponding items and forwards them to Bob. Otherwise, it sets state variable as **recovered**, stores  $(\ell, S_B(P, \ell, EK), EB, r_2)$  into its database, forwards  $(\ell, S_B(P, \ell, EK))$  and  $(\ell, EB, r_2)$  to Alice and Bob, respectively.

Note that in the above scheme, timeliness is achieved since both Alice and Bob can terminate a protocol instance unilaterally at any moment before the deadline  $t$ . Different from other schemes supporting timeliness [18,10,13,12], however, deadline  $t$  is used in our scheme to support *weak* stateless TTP. That is, after time  $t$  the TTP could remove all state information related to deadline  $t$  into a log system. Therefore, the TTP only needs to maintain a relatively small searching database.

**Dispute Resolution Policy II.** This policy is almost the same as the Dispute Resolution Policy I, except the following difference. When the judge deals with non-repudiation of receipt, it further needs to inquire Bob or the TTP whether they could provide abort token  $S_A(P, \ell, \text{abort})$ . If this is true, the

judge dismisses Alice’s request. On the other hand, if all evidences are correct and no valid abort token is provided, the judge accepts Alice’s claim.

Note that, we need to make the above change in our Dispute Resolution Policy II. Otherwise, Alice could cheat Bob as follows. Upon receiving Bob’s signature  $S_B$ , Alice does not reveal  $(EB, r_2)$  to Bob. At the same time, she aborts the protocol run by contacting the TTP so that Bob cannot get  $(EB, r_2)$  from the TTP. According our new policy, however, in such case Bob can provide abort token to counteract the power of  $S_B$  so that the result is still fair for both parties. Moreover, in the abort protocol, it is also necessary to encrypt abort token  $S_A(P, \ell, \text{abort})$  under the TTP’s public key. Otherwise, the following unfair situation may happen. After getting message flow (e1), Bob directly obtains  $(EB, r_2)$  from the TTP by successfully executing the recovery protocol. However, Alice may initiate the abort protocol almost simultaneously, since she does not receive Bob’s signature. By eavesdropping the communication between Alice and the TTP, it is possible that Bob gets both  $(EB, r_2)$  and the abort token. So, Bob can access message  $m$  by using  $(EB, r_2)$ , and deny that he already obtained  $m$  by providing abort token.

*Remark 3.* For simplicity, the same symbol  $P$  is used to denote the two protocol identifiers in the above two schemes. However, according our assumption, protocol identifier is unique for each protocol, so we actually need to differentiate them by using two distinct symbols, e.g.,  $P_1$  and  $P_2$ .

## 5 Security Discussion

In this section, we only argue that our second certified email scheme guarantees the fairness, the most important security requirement for all fair-exchange protocols. There are two reasons for this arrangement: (1) It is easy to see that other properties are satisfied as we claimed in Table 1; and (2) Fairness is also guaranteed in our first scheme by a similar but simpler argument. Intuitively, our schemes support fairness since both  $S_A(P, \ell, EK)$  and  $S_B(P, \ell, EK)$  can be explained as valid NRO and NRR evidences *if and only if* all related values ( $c, HC, HK, EB, EK, t$  etc.) are prepared correctly. We classify our discussion into two cases: (1) Alice is honest, but Bob is trying to cheat; and (2) Bob is honest, but Alice is trying to cheat.

**Case 1:** *Alice is honest, but Bob is trying to cheat.* Alice is assumed to be honest now, so message flow (e1) is correctly prepared. When Bob receives such a valid message flow (e1), he has to ask himself whether he wants to access the encrypted message. If not, he could ignore this protocol run without loss fairness. If yes, Bob has to get  $(EB, r_2)$  at first. However,  $EB$  is securely encrypted under the TTP’s public key, and prepared by Alice, so only Alice or the TTP can reveal the pair  $(EB, r_2)$  to Bob. Furthermore, according to the protocol specification, to get  $(EB, r_2)$  Bob is required to send his signature  $S_B(P, \ell, EK)$  to Bob or the TTP before deadline  $t$  and under the condition that this protocol run is not aborted yet. In particular, this implies that Bob or anybody else cannot get  $(EB, r_2)$  by sending  $S_B(P, \ell', EK)$  or  $S_{B'}(P, \ell', EK)$  (and related information)

to the TTP. Because in such cases the TTP will find the first part of decrypted content of  $EK$  does not match label  $\ell$ . Therefore, Bob cannot access the message or get valid NRO evidence without issuing receipt. That is, our scheme is fair for honest initiator (Alice).

**Case 2:** *Bob is honest, but Alice is trying to cheat.* The purpose of dishonest sender Alice is to get valid NRR evidence (i.e.,  $S_B(P, \ell, EK)$ ) from Bob so that Bob cannot access the message  $m$  or cannot get valid NRO evidence (i.e.,  $S_B(P, \ell, EK)$ ). However, this actually implies that Alice has to prepared message (e1) correctly. Otherwise, even if she got  $S_B(P, \ell, EK)$  from Bob, this signature cannot be explained as valid NRR evidence, due to some inconsistency among  $m, c, HC, HK, EB, EK, t$ . The reason is that in our protocol,  $S_B(P, \ell, EK)$  can be explained as valid NRR evidence showing that Bob indeed received message  $m$  from Alice, *if and only if* all the following conditions hold:

- 1) Alice can provide  $(P, A, B, T, m, k, r_1, r_2, t, S_B(P, \ell, EK))$  such that  $c = E_k(m)$ ,  $EB = E_B^{r_1}(k)$ ,  $\ell = H(P, A, B, T, H(c), H(k), t)$ ,  $EK = E_T^{r_2}(\ell, EB)$ , and  $S_B(P, \ell, EK)$  is Bob's signature on  $(P, \ell, EK)$ .
- 2) Bob cannot provide valid abort token  $S_A(P, \ell, \text{abort})$ .

Hence, the last cheating strategy for Alice is to thwart Bob getting the pair  $(EB, r_2)$  after she obtained Bob's receipt. Simply refusing to reveal  $(EB, r_2)$  to Bob is essentially unharmed to Bob, since the latter can execute the recovery protocol and then get this pair directly from the TTP before the deadline  $t$ . At the same time, as we mentioned after the description of our policy II, Alice also cannot achieve her goal by initiating the abort protocol as soon as she received Bob's signature  $S_B(P, \ell, EK)$ . The reason is that in this situation, the only two possible outputs are both fair: (a) This protocol run has been recovered, so Bob already received the  $(EB, r_2)$ ; or (b) This protocol will be aborted by the TTP, so Bob will get the abort token  $S_A(P, \ell, \text{abort})$ , which is supposed to annul Bob's receipt  $S_B(P, \ell, EK)$ . Therefore, once again, our protocol does not enable the cheater (Alice) taking an advantage over the honest party (Bob). That is, our certified email scheme is fair for honest receiver.

## 6 Conclusion

In this paper, we first pointed out some weaknesses in several existing certified email schemes proposed in [9,14]. After that, we proposed two new generic optimistic schemes for certified email delivery. As Table 1 demonstrated, our protocols provide as many as possible desirable properties. At the same, the proposed solutions achieve better performance than the-state-of-the-art protocols in this field. Specifically, our schemes overcome the security shortcoming in schemes [9,14]; support transparent TTP while the schemes in [10,12] do not; provide evidences for non-repudiation of origin while the scheme in [3] does not; and satisfy timeliness while the schemes in [3,9] do not.



## References

1. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In: *Proc. of AMC Conference on Computer and Communications Security (CCS'97)*, pp. 7-17. ACM Press, 1997.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18 (4): 591-606, 2000.
3. G. Ateniese and C. Nita-Rotaru. Stateless-receipt certified E-mail system based on verifiable encryption. In: *CT-RSA '02*, LNCS 2271, pp. 182-199. Springer-Verlag, 2002.
4. G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signature. In: *Proc. of AMC Conference on Computer and Communications Security (CCS'99)*, pp. 138-146. ACM Press, 1999.
5. F. Bao, R.H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In: *Proc. of IEEE Symposium on Security and Privacy*, pp. 77-85. IEEE Computer Society, 1998.
6. F. Bao, G. Wang, J. Zhou, and H. Zhu. Analysis and improvement of Micali's fair contract signing protocol. In: *Information Security and Privacy (ACISP'04)*, LNCS 3108, pp. 176-187. Springer-Verlag, 2004.
7. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: *Crypto'98*, LNCS 1462, pp.13-25. Springer-Verlag, 1998.
8. R. Deng, L. Gong, A. Lazar, and W. Wang. Practical protocol for certified electronic mail. *Journal of Network and Systems Management*, 1996, 4(3):279-297.
9. K. Imamoto and K. Sakurai. A certified e-mail system with receiver's selective usage of delivery authority. In: *Indocrypt 2002*, LNCS 2551, pp. 326-338. Springer-Verlag, 2002.
10. S. Kremer and O. Markowitch. Selective receipt in certified e-mail. In: *Indocrypt 2001*, LNCS 2247, pp. 136-148. Springer-Verlag, 2001.
11. S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17): 1606-1621. Elsevier, Nov. 2002.
12. S. Gürgens, C. Rudolph, and H. Vogt. On the security of fair non-repudiation protocols. In: *Information Security Conference (ISC'03)*, LNCS 2851, pp. 193-207. Springer-Verlag, 2003.
13. O. Markowitch and S. Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In: *Information Security Conference (ISC'01)*, LNCS 2200, pp. 363-378. Springer-Verlag, 2001.
14. S. Micali. Simple and fast optimistic protocols for fair electronic exchange. In: *Proc. of 22th Annual ACM Symp. on Principles of Distributed Computing (PODC'03)*, pp. 12-19. ACM Press, 2003.
15. V. Shoup. OAEP reconsidered. *Journal of Cryptology*, 15(4): 223-249, 2002.
16. J. Zhou and D. Gollmann. Certified electronic mail. In: *Computer Security - ES-ORICS'96*, LNCS 1146, pp. 160-171. Springer-Verlag, 1996.
17. J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In: *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97)*, pp. 126-132. IEEE Computer Society Press, 1997.
18. J. Zhou, R. Deng, and F. Bao. Evolution of fair non-repudiation with TTP. In: *Information Security and Privacy (ACISP'99)*, LNCS 1587, pp. 258-269. Springer-Verlag, 1999.

# Cryptanalysis of a Forward Secure Blind Signature Scheme with Provable Security

Shuhong Wang<sup>1</sup>, Feng Bao<sup>2</sup>, and Robert H. Deng<sup>1</sup>

<sup>1</sup> School of Information Systems, SMU, Singapore 178902  
{shwang, robertdeng}@smu.edu.sg

<sup>2</sup> Institute for Infocomm Research (I2R), Singapore 119613  
baofeng@i2r.a-star.edu.sg

**Abstract.** A forward secure blind signature scheme was proposed by Duc, Cheon and Kim, in ICICS 2003. The security of the scheme was proved to be equivalent to the strong RSA assumption in the random oracle model. In this paper we present an attack to the scheme by forging valid signatures with public keys only. The attack is so efficient that forging a valid signature needs less computation than legally generating a signature, even considering only the user side. Our result implies that the security proof of the scheme must be invalid. Furthermore we point out the fault of the proof and explain why it invalidates the proof.

**Keywords:** Blind signature, Forward security, Provable security, Strong RSA assumption, Cryptanalysis.

## 1 Introduction

Due to some unpredictable security faults of underlying system or errors of implementation, key exposure is high likely unavoidable. To mitigate the danger caused by key exposure, the notion of *forward security*<sup>1</sup> was introduced [1] in the context of signature schemes. It is always obtained by employing the so called key evolution strategy, which is economical/practical compared with distribution of keys across multiple systems via secret sharing like threshold methodologies [9,10]. Informally, key evolution means that different secret keys are used in different periods of time, and the key for next time period is updated (through an **Update** protocol) from the one in previous time period, meanwhile the public key is kept unchanged over its lifetime. There have been many signature schemes with forward security [4,2,13].

However, as claimed in [8], there is no instance of **Update** supporting unlimited periods<sup>2</sup> key evolution until the proposal of Duc, Cheon and Kim [8] in a

---

<sup>1</sup> In the context of session key exchange, it was first introduced in [11] known as forward secrecy, meaning that compromise of the current session key should not compromise past established session keys.

<sup>2</sup> In general, the **Update** protocols only support  $T$  times of key evolutions for some predefined integer  $T$ .

*blind signature* context. The scheme is called forward secure blind signature, and hereinafter, we denote it the DCK scheme for short.

Blind signature, as an extension of digital signature, allows the user to obtain the signer’s signature on a message of his choice in a blind way such that the message content is not revealed to the signer. Such a scheme was first proposed by Chaum [6] for the purpose of digital payments where the user is a consumer and the signer is a bank. Along with the rapid development of sensitive e-commerce [16,3,18], blind signature is becoming very concernful. It is unassailable that forward security will provide really useful features for a blind signature scheme.

Desirably, a *forward secure blind signature* [8], especially when used for electronic payment, should at least have following two security features.

- **Blindness.** Besides “obtaining signature without revealing message”, the blindness property also implies that the signer cannot statistically distinguish signatures, which is like that the bank cannot trace its client’s buying activities. For the formal definition, please refer to [8].
- **Forward security.** In the context of blind signature, forward security implies the basic *unforgeability* as of ordinary signatures. In addition, it implies the unforgeability of signature to be valid in previous time periods even if the current secret key of the signer is compromised.

In this paper, we address the security analysis of the attractive DCK scheme [8] mentioned above. Although the security of the scheme was proved to be equivalent to the strong RSA assumption in the random oracle model [5], we are still able to forge valid signatures at will with public keys only. Note that our attack is so efficient that forging a valid signature needs less computation than legally generating a signature, even when merely considering the user side. Our result implies that the security proof of the scheme must be invalid. Furthermore we point out the fault of the security proof and explain why it invalidates the proof.

The remainder of the paper is arranged as follows. We first briefly review the original DCK scheme in section 2, and then describe our signature forgery attack in section 3. The analysis of their security proof is given in section 4, followed by conclusion in section 5.

## 2 Description of the DCK Scheme

The DCK scheme is simply described as follows.

**System Setup.**  $k$  is the security parameter.  $N = pq$  are product of two random safe primes  $p$  and  $q$  of  $k/2$  bits length.  $\lambda$  is a large prime without nontrivial common divisor with  $\varphi(N)$ , where  $\varphi$  is the Euler function in number theory. An element  $a \in Z_N^*$  is selected to be of order greater than  $\lambda$ . Let  $r_0 \in_R Z_\lambda^*$  and  $s_0 \in_R Z_N^*$  (with notation  $x \in_R X$ , we mean  $x$  is randomly chosen from  $X$ ), and compute  $V = a^{-r_0} s_0^{-\lambda} \bmod N$ .

Finally, the signer's initial secret key  $SK_0$  is  $(0, r_0, s_0, v_0 = V, f_0 = 1)$ , the public key  $PK = (N, \lambda, a, V)$ . All other parameters are erased. Also, a collision-free hash function  $H : \{0, 1\}^* \rightarrow Z_\lambda^*$  is assumed and made public.

**Secret key Update.** Hereafter, we define  $a \div b$  the integer quotient of  $\frac{a - (a \bmod b)}{b}$ . After each execution of the **Update** protocol, all parameters except the new secret key are erased from the memory. See Fig. 1.

Input $SK_i$	$e \in_R Z_N^*$	Output $SK_{i+1}$
$(i, r_i, s_i, f_i) \Rightarrow$	$f_{i+1} = f_i^2 a^e \bmod N$	$\Rightarrow (i+1, r_{i+1}, s_{i+1}, f_{i+1})$
	$l = (2r_i - e) \div \lambda$	
	$r_{i+1} = (2r_i - e) \bmod \lambda$	
	$s_{i+1} = a^l s_i^2 \bmod N$	

**Fig. 1.** The key Update protocol

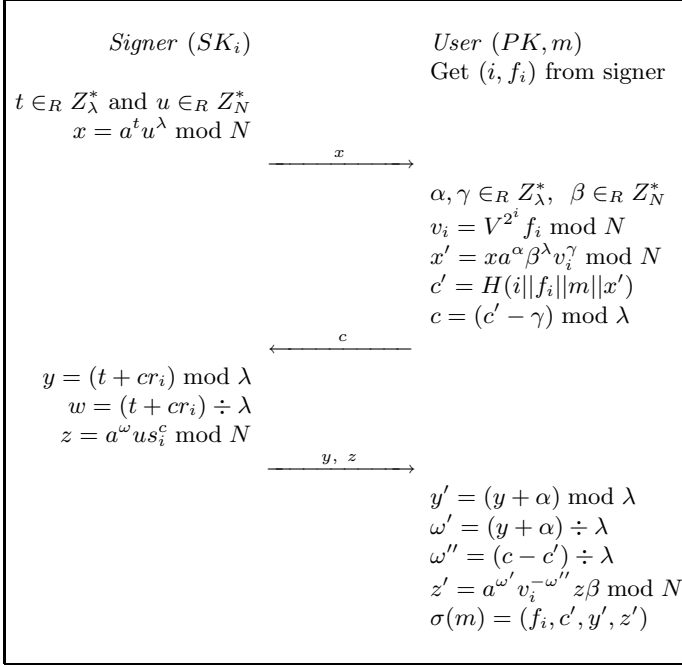
**Signature Issuing.** This procedure involves two entities: signer and user. The protocol is illustrated in Fig. 2, where  $\parallel$  denotes the string concatenation. Note that  $(i, f_i)$  are available to the user when contacting with the signer, but they are unavailable to verifier. Otherwise, it contradicts the original intention for offline electronic system, where payments are made available without online communication with the signer (bank) [8].

**Signature Verification.** Given the signature  $(m, i, \sigma(m)) = (m, i, f, c', y', z')$  and  $PK = (N, \lambda, a, V)$ , the verifier first computes  $v_i = V^{2^i} f \bmod N$  and  $x'' = a^{y'} z'^{\lambda} v_i^{c'}$ , then checks whether or not  $c' \stackrel{?}{=} H(i \parallel f \parallel m \parallel x'')$ . If it is right, then accept; Otherwise reject.

### 3 The Signature Forgery Attack

In this section we describe our forgery attack on the DCK scheme. The attack is so strong that anyone can forge signatures on any message valid in any time period. The only needed information is the public key  $PK = (N, \lambda, a, V)$ . The attack is also very efficient even compared with the computation on the user side only. The attack behaves as the following.

1. Obtain the public key  $PK = (N, \lambda, a, V)$  and a valid time period  $i$ .
2. Choose  $\alpha, \gamma \in_R Z_\lambda^*$  and  $\beta, v \in_R Z_N^*$ .
3. Compute  $f = V^{-2^i} v^\lambda \bmod N$ .
4. Compute  $x = a^\alpha \beta^\lambda v^{\lambda \gamma} \bmod N$ .
5. Compute  $c = H(i \parallel f \parallel m \parallel x)$ , where  $m$  is any message of the attacker's choice.
6. Compute  $z = v^{\gamma - c} \beta \bmod N$  and set  $y = \alpha$ .
7. Output the 6-tuple  $(m, i, f, c, y, z)$  as the forged signature on  $m$ , intending for the  $i$ -th time period.



**Fig. 2.** The signature issuing protocol. As noted in [8], the index  $i$  of  $f_i$  is omitted in  $\sigma(m)$  on consideration that attackers do not have to use the correct  $f$  for a period. Thus the signature is denoted as  $(m, i, \sigma(m)) = (m, i, f, c', y', z')$ .

**Theorem 1. (Correctness of the Attack)** Suppose an attacker follows above seven steps and obtains the 6-tuple  $(m, i, f, c, y, z)$ . Then, the verifier always accepts  $(m, i, f, c, y, z)$  as a valid signature in period  $i$ .

*Proof.* To prove the theorem, we simply simulate what the verifier does with the signature  $(m, i, f, c, y, z)$ . He/She first retrieves the public key  $PK = (N, \lambda, a, V)$  and computes  $v_i = V^{2^i} f = V^{2^i} (V^{-2^i} v^\lambda) = v^\lambda \bmod N$ . Then he/she computes  $x'' = a^y z^\lambda v_i^c \bmod N$ . Because we have

$$\begin{aligned}
 x'' &= a^y z^\lambda v_i^c \bmod N \\
 &= a^y (v^{\gamma-c} \beta)^\lambda v_i^c \bmod N \\
 &= a^\alpha v^{(\gamma-c)\lambda} \beta^\lambda (v^\lambda)^c \bmod N \\
 &= a^\alpha \beta^\lambda v^{(\gamma-c)\lambda + c\lambda} \bmod N \\
 &= a^\alpha \beta^\lambda v^{\gamma\lambda} \bmod N = x,
 \end{aligned}$$

it is always the case that  $c = H(i || f || m || x) = H(i || f || m || x'')$ . As a result, the verifier accepts  $(m, i, f, c, y, z)$  as a valid signature in period  $i$ . ■

**Remarks on Efficiency.** Roughly speaking, our forgery attack only expends  $6 T_E$  (modulo exponentials) and  $4 T_M$  (modulo multiplications), while there are 5

$T_E$  and 6  $T_M$  on only the user side for legally obtaining a signature. Both have a modulo reciprocal and therefore are eliminated. Note that we do not count in the computation of  $v_i = V^{s^i} f_i \bmod N$  which is assumed to be available for legal users from the signer. In fact, the attack can also contact with the signer and get the  $V^{-2^i}$  by computing  $f_i v_i^{-1} \bmod N$ , thus 1  $T_M$  and 1 reciprocal replace 1  $T_E$  and 1 reciprocal. Accordingly, the forgery gains 1  $T_M$  efficiency compared to honestly obtaining a signature by the user (5  $T_E$  and 5  $T_M$  for forgery to 5  $T_E$  and 6  $T_M$  for generation).

## 4 The Failure of Security Proof

The DCK scheme [8] is constructed from the provably secure Okamoto-Guilou-Quisquater (OGQ for short) blind signature scheme [15,12]. Using the same methodology (oracle replay) for proving OGQ scheme due to Pointcheval and Stern [16], authors of [8] proved the security of DCK scheme under the strong RSA assumption.

**Strong RSA Assumption.** The strong RSA assumption is described as follows: Given a RSA modulus  $N$  (which is a product of two large primes) and a random element  $c \in Z_N^*$ , it is intractable to find two elements  $m, r \in Z_N^*$  such that  $m^r = c \bmod N$ . It is a well-known assumption in cryptography and has been extensively used for security proofs.

### 4.1 Sketch of the Security Proof

There are two theorems regarding the security of the DCK scheme, one for the blindness property (Theorem 2 of [8]) and one for the forward security (Theorem 3 of [8]). The later is outlined as follows.

**Theorem 3 of [8].** *If there exists a forger who can break forward security of our scheme. Then, with non-negligible probability, we can violate the strong RSA assumption.*

*Proof outline.* Assume a forger  $\mathcal{F}$  who obtains  $PK$  and  $SK_i$  of time period  $i$  can output a signature  $\sigma(m)$  valid at some time period  $j$  for  $j < i$ . Also assume  $\mathcal{F}$  should query the hashing oracle on  $(j||f||m||x')$  before its output. Upon the answer of the oracle say  $H_1$ ,  $\mathcal{F}$  successfully forge a signature  $(j, \sigma_1(m)) = (j, f, x'_1, y'_1, z'_1)$ . Then, by replaying another oracle  $H_2$  which has the same answer to oracle  $H_1$  until the query of  $(j||f||m||x')$ . With non-negligible probability,  $\mathcal{F}$  will again output a forged signature  $(j, \sigma_2(m)) = (j, f, x'_2, y'_2, z'_2)$  based on oracle  $H_2$ , this is assured by the well-known forking lemma [16]. Since the two forged signatures have the same verifying equation, it must be the case that  $a^{y'_1} z'_1{}^\lambda (V^{2^j} f)^{c'_1} = a^{y'_2} z'_2{}^\lambda (V^{2^j} f)^{c'_2}$ . By assuming  $V^{2^j} f$  equals to  $v_j$  and therefore has the form of  $a^{-r_j} s_j^{-\lambda} \bmod N$ , the authors of [8] claimed being able to come up with an equation of the form  $a^\rho = b^\lambda \bmod N$ , and thus the strong RSA problem is solvable with a high probability, if only  $\gcd(\rho, \lambda) = 1$  (see Lemma 1 of [8]). Note that it is high likely  $\gcd(\rho, \lambda) = 1$  with  $\lambda$  being prime.

## 4.2 Fault of the Proof

As above mentioned, our result implies that the security proof of the scheme must be invalid. Although there exists negative examples [7] such that schemes provably secure in random oracle model [5] may result in insecure ones when the oracle is implemented by cryptographic hash functions, our attack has nothing to do with the hash function. In fact, it is on the basis of the problem in the scheme construction itself.

Keeping our attack in mind to check through the security proof, it is not hard to find out its fault. For a forged signature, the expectation [8] that  $V^{2^j} f \bmod N$  would equal to the correct  $v_j = a^{-r_j} s_j^{-\lambda} \bmod N$  as in the Update protocol is unreliable. In the proposed attack, we have  $V^{2^j} f = v^\gamma \bmod N$  with  $\gamma \in_R Z_\lambda^*$  and  $v \in_R Z_N^*$ , clearly it is not in the form of  $a^r s^\lambda \bmod N$  for some  $r \in Z_\lambda^*$  and  $s \in Z_N^*$ .

In the following, we show how critical the fault is. To equalize the security of DCK scheme and the strong RSA assumption, it is sufficient to get an equation like  $a^\rho = b^\lambda \bmod N$ . Let us take an observation on the equation obtained by oracle replay:  $a^{y'_1} z_1'^{\lambda} (V^{2^j} f)^{c'_1} = a^{y'_2} z_2'^{\lambda} (V^{2^j} f)^{c'_2}$ , which can be transferred to  $a^{y'_1 - y'_2} = (z_2'/z_1')^\lambda \cdot (V^{2^j} f)^{c'_2 - c'_1} \bmod N$ . Obviously, one can get equation  $a^\rho = b^\lambda \bmod N$  with some  $\rho \in Z$  and  $b \in Z_N^*$  if and only if  $(V^{2^j} f)^{c'_2 - c'_1} \bmod N$  can be expressed as  $a^r s^\lambda \bmod N$ . However, unless with negligible probability  $c'_2 = c'_1 \bmod \lambda$  (then,  $(V^{2^j} f)^{c'_2 - c'_1} = [(V^{2^j} f)^{(c'_2 - c'_1) \div \lambda}]^\lambda \bmod N$ ), being able to express an random elements in  $Z_N^*$  as the form of  $a^r s^\lambda \bmod N$  means that one can easily break the OGQ scheme [15] by just using  $(\lambda - r, as)$  as an OGQ signing key pair ( $a^r s^\lambda = a^{-(\lambda - r)} (as)^\lambda \bmod N$ ). This result contradicts the proof of Pointcheval and Stern [16]. And if the proof in [16] is correct, expressing an random element in  $Z_N^*$  as  $a^r s^\lambda$ , itself is at least as hard as the strong RSA problem. In other words, the authors in [8] implicitly assumed the solvability of the strong RSA problem. Assume a problem has already been solved and then turn back to solve that problem, which is logically incorrect.

**Remarks.** However, the proof of [8] still implies the security of the Update protocol, i.e., it is impossible for an attacker to forge a signature by forging the secret key in advance. Since in this case, the equation  $V^{2^j} f = v_j = a^{-r_j} s_j^{-\lambda} \bmod N$  holds.

## 5 Conclusion

In this paper, we successfully illustrated the insecurity of a forward-secure blind signature scheme which is proved to be equivalent to the strong RSA assumption. The attack is strong and very efficient. Anyone can forge signatures on any message valid in any time period using the unchanged public keys only. Furthermore, we also pointed out the fault of the security proof and explained why it invalidates the proof. Our work implies that regardless of the failure caused by oracle implementations, the security proof itself still needs time to validate its correctness.

## References

1. Ross Anderson, Two Remarks on Public Key Cryptography, Invited Lecture, in *Fourth Annual Conference on Computer and Communications Security*, ACM, 1997.
  2. Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2000*, Springer-Verlag, 2000.
  3. Feng Bao, Robert H. Deng, and Wenbo Mao, Efficient and practical fair exchange protocols with off-line TTP, in *IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp. 77 - 85, 1998.
  4. Mihir Bellare and Sara K. Miner, A Forward-Secure Digital Signature Scheme, in *Advances in Cryptology - CRYPTO '99*, LNCS 1666, Springer-Verlag, pp. 431 - 448, 1999.
  5. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM CCS '93*, pages 62 - 73, November 1993.
  6. David Chaum, Blind Signatures For Untraceable Payments, in *Advances in Cryptology - CRYPTO '82*, Plenum Publishing, pp. 199 - 204, 1982.
  7. Ran Canetti, Oded Goldreich, and Shai Halevi, The Random Oracle Methodology, Revisited (Extend abstract), in *Proc. of the 30th ACM Symp. on Theory of Computing - STOC'98*, pages 209-218, 1998.
  8. Dang N. Duc, Jung H. Cheon, and Kwangjo Kim, A Forward-Secure Blind Signature Scheme Based on the Strong RSA Assumption, in *Proceedings of the 5-th International Conference on Information and Communications Security - ICICS '03*, LNCS 2836, Springer-Verlag, pp. 11 - 21, 2003.
  9. Yvo G. Desmedt and Yair Frankel, Threshold cryptosystems, in *Advances in Cryptology - Crypto '89*, LNCS 435, Springer-Verlag, pp. 307 - 315, 1989.
  10. Y. Desmedt, Y. Frankel and M. Yung, Multi-receiver/Multi-sender network security: efficient authenticated multicast/feedback, *Proceedings of IEEE Infocom '92*, pp. 2045 - 2054, 1992.
  11. C. Günther, An Identity-based Key-exchange Protocol, in *Proceedings of Eurocrypt '89*, LNCS 434, Springer-Verlag, 1989.
  12. Louis S. Guillou and Jean J. Quisquater, A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing both Transmission and Memory, in *Advances in Cryptology - EUROCRYPT '88*, LNCS 330, Springer-Verlag, pp. 123 - 128, 1988.
  13. Gene Itkis and Leonid Reyzin, Forward-Secure Signatures with Optimal Signing and Verifying, in *Advances in Cryptology - CRYPTO '01*, LNCS 2139, Springer-Verlag, pp. 332 - 354, 2001.
  14. Wenbo Mao and Colin Boyd, Towards Formal Analysis of Security Protocols, In *Proceedings of the 4-th Computer Security Foundations Workshop*, Franconia, New-Hampshire, June 1993.
  15. Tatsuki Okamoto, Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes, in *Advances in Cryptology - CRYPTO '92*, LNCS 740, Springer-Verlag, pp. 31 - 53, 1992.
  16. David Pointcheval and Jacques Stern, Security Arguments for Digital Signatures and Blind Signatures, *Journal of Cryptology*, Vol. 13(3), pp. 361 - 396, Springer-Verlag, 2000.
- The full version of the authors' "Security proofs for Signature Schemes" in Eurocrypt '96 and "Provably Secure Blind Signature Schemes" in Asiacypt '96.



17. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, How to share a function securely, in *Proceedings of 26th STOC*, pp. 522 - 533, 1994.
18. S. Wong and Victor K. Wei, A method for imposing spending limit on electronic coins, in *Proceedings of Int'l Symp. on Information Theory*, 1998.
19. Fanguo Zhang and Kwangjo Kim, ID-Based Blind Signature and Ring Signature from Pairings, in *Advances in Cryptology - ASIACRYPT '02*, LNCS 2501, Springer-Verlag, pp. 533 - 547, 2002.

# On Delegatability of Four Designated Verifier Signatures

Yong Li<sup>1</sup>, Helger Lipmaa<sup>2,3</sup>, and Dingyi Pei<sup>1</sup>

<sup>1</sup> State Key Laboratory of Information Security (Graduate School of Chinese Academy of Sciences), Beijing 100049, P.R. China

<sup>2</sup> Cybernetica AS, Lai 6, 51005 Tartu, Estonia

<sup>3</sup> Institute of Computer Science, University of Tartu, J. Liivi 2, 50409 Tartu, Estonia

**Abstract.** In a paper recently published in ICALP 2005, Lipmaa, Wang and Bao identified a new essential security property, non-delegatability, of designated verifier signature (DVS) schemes. Briefly, in a non-delegatable DVS scheme, neither a signer nor a designated verifier can delegate the signing rights to any third party  $T$  without revealing their secret keys. We show that the Susilo-Zhang-Mu identity-based strong DVS scheme, Ng-Susilo-Mu universal designated multi verifier signature scheme, the Laguillaumie-Vergnaud multi-DVS scheme and the Zhang-Furukawa-Imai universal DVS scheme are delegatable. Together with the results of Lipmaa, Wang and Bao, our results show that most of the previously proposed DVS schemes are delegatable. However, the Laguillaumie-Vergnaud and Zhang-Furukawa-Imai schemes may still be secure in practice, since there the only party who can delegate signing is the designated verifier, who may not have motivation to do so. We finish the paper with some discussion on whether the non-delegatability notion of Lipmaa, Wang and Bao is appropriate.

**Keywords:** Designated verifier signatures, non-delegatability.

## 1 Introduction

A designated verifier signature (DVS) scheme [JSI96, Cha96] enables a signer to sign a message so that the designated verifier can verify it as coming from the signer. However, the designated verifier cannot transfer the conviction to others because he himself is able to generate signatures according to a distribution that is computationally or statistically close to the distribution of signatures, generated by the signer. On the other hand, nobody else but the signer and the designated verifier can generate valid signatures.

Recently, Lipmaa, Wang and Bao revisited the DVS security in [LWB05]. In particular, they identified a new security property for DVS, non-delegatability, and showed that several previously proposed DVS schemes [SKM03, SBWP03, SWP04, LV04a] are delegatable. Informally speaking, DVS is delegatable if either the signer or the designated verifier can delegate the signing rights (either with respect to a concrete designated verifier or with respect to all designated verifiers) to some third party  $T$  without disclosing his or her secret key. Delegatability, especially with respect to a concrete designated verifier, is highly undesirable in many applications. For example, in an e-voting scenario where a voter signs messages by using a delegatable DVS scheme (with the tallier being the designated verifier), one voter can delegate her voting right to a coercer that can then vote instead of the voter. Therefore, such an e-voting protocol is

coercible. Moreover, in many e-commerce applications, one can use a DVS scheme so that the signer is a subscriber to an e-service provided by a service provider who is the designated verifier. If the DVS scheme is delegatable, signer can send some delegation token to a non-subscriber who can then enjoy the service for free.

**Our Contributions.** In addition to the negative results of [LWB05], we show that the Susilo-Zhang-Mu ID-based DVS scheme SZM04 [SZM04], the Ng-Susilo-Mu universal designated multi verifier signature scheme NSM05 [NSM05], the Zhang-Furikawa-Imai DVS scheme ZFI05 [ZFI05] and the Laguillaumie-Vergnaud MDVS scheme LV04 [LV04b] are delegatable. Together with [LWB05], our results show that almost all DVS schemes in the literature are delegatable. In particular, all DVS schemes based on bilinear maps are delegatable. The only non-delegatable DVS schemes that we are aware of are the schemes from [JSI96, LWB05] that are both built by using standard proof of knowledge techniques.

All delegation attacks, proposed in [LWB05], had a similar structure: they showed that either the signer or the designated verifier can delegate the signing rights by publishing some Diffie-Hellman key. Our attacks are more varied. In particular, our attacks against ZFI05 and LV04, while being attacks according to the definitions of Lipmaa, Wang and Bao, might sometimes be not very serious in practice. Namely, in both cases, only the designated verifier (in the case of LV04, the coalition of two designated verifiers) can delegate the signing. This means that attacks in the scenarios, outlined above, will not work. However, there are still some possibilities of cheating. For example, the service provider can forward the delegation token to a third party, who can then use the service indistinguishably from the real signer. By doing so, the third party could act in a way that ruins the reputation of the real signer. Whether this is a real attack or not, depends on the situation; we will leave it as an open question. For applications where this is a real attack, one should not use ZFI05 and LV04. In applications where it is not, one should give an alternative and possibly weaker definition of non-delegatability than was done in [LWB05].

Inspired on this, we give an informal definition of a weaker notion of delegatability that we call *verifier-only delegatability*. Intuitively, a (multi-)DVS scheme is verifier-only delegatable if the designated verifier (but not the signer) can delegate the signing rights to some third party. Clearly, a verifier-only delegatable DVS scheme is also delegatable. It seems (seems, since we are not going to give proofs that the signer cannot delegate) that the LV04 and the ZFI05 schemes are verifier-only delegatable.

Moreover, the presented attacks can also be divided into delegation attacks that allow to delegate the signing rights of a fixed signer w.r.t. a fixed tuple of designated verifiers, or the rights of *any* signer w.r.t. a fixed tuple of designated verifiers, or the rights of a fixed signer w.r.t. any tuple of designated verifiers. According to [LWB05], existence of any of these attacks makes a scheme delegatable. Again, it can be argued that the first type of attack is the most serious one since the last two attack types give too much power to the third party  $T$  (and therefore one might be less motivated to delegate the rights). One can try to modify the delegatability definition so that only the first attack type is classified as an attack. We will leave it as an open question whether this is reasonable.

Regardless of the previous comments, our own opinion is that our attacks against all four schemes indicate some weaknesses in them and while the non-delegatability

definition of [LWB05] might be too strong in some sense, to avoid any kind of future attack and unexpected vulnerabilities, it is a good idea to design DVS schemes that are non-delegatable according to [LWB05].

**Road-map.** Formal definition of an  $n$ -DVS scheme and its security is given in Sect. 2. In Sect. 3, we review four DVS schemes and present our delegation attacks against every single one of them. We discuss the different delegation attacks and define the novel notion verifier-only non-delegatability in Sect. 4.

## 2 Preliminaries

Let  $\mathbb{G}$  be a cyclic additive group generated by  $P$ , whose order is a prime  $q$ , and let  $\mathbb{H}$  be a cyclic multiplicative group of the same order  $q$ . A bilinear pairing is a map  $\langle \cdot, \cdot \rangle : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{H}$  with the following properties:

**Bilinearity:**  $\langle aP, bQ \rangle = \langle P, Q \rangle^{ab}$  for all  $P, Q \in \mathbb{G}$  and  $a, b \in \mathbb{Z}_q^*$ ;

**Non-degeneracy:** There exist  $P, Q \in \mathbb{G}$  such that  $\langle P, Q \rangle \neq 1$ ;

**Computability:** There is an efficient algorithm to compute  $\langle P, Q \rangle$  for all  $P, Q \in \mathbb{G}$ .

**Formal Definition of  $n$ -DVS.** Next, we present a formal definition of  $n$ -DVS for  $n \geq 1$ , generalising the definition from [LWB05]. Let  $S$  be the signer, and  $D_1, \dots, D_n$  be  $n$  designated verifiers. In the following, we will denote  $(PK_{D_1}, \dots, PK_{D_n})$  by  $PK_D$ ,  $(SK_{D_1}, \dots, SK_{D_n})$  by  $SK_D$ , and  $(\text{Simul}_{PK_S, PK_D, SK_{D_1}}, \dots, \text{Simul}_{PK_S, PK_D, SK_{D_n}})$  by  $\text{Simul}_{PK_S, PK_D, SK_D}$ .

Let  $\mathcal{M}$  be the message space. Given a positive integer  $n$ , an  $n$ -designated verifier signature ( $n$ -DVS) scheme is defined by the following algorithms:

- Setup is a probabilistic algorithm that outputs the public parameter  $param$ ;
- KeyGen( $param$ ) is a probabilistic algorithm that takes the public parameters as an input and outputs a secret/public key-pair  $(SK, PK)$ ;
- Sign $_{SK_S, PK_D}(m)$  takes as inputs signer's secret key, designated verifiers' public keys, a message  $m \in \mathcal{M}$  and a possible random string, and outputs a signature  $\sigma$ ;
- For  $i \in [1, n]$ , Simul $_{PK_S, PK_D, SK_{D_i}}(m)$  takes as inputs signer's public key, designated verifiers' public keys, secret key of one designated verifier, a message  $m \in \mathcal{M}$  and a possible random string, and outputs a signature  $\sigma$ ;
- Verify $_{PK_S, PK_D}(m, \sigma)$  is a deterministic algorithm that takes as inputs a signing public key  $PK_S$ , public keys of all designated verifiers  $D_i, i \in [1, n]$ , a message  $m \in \mathcal{M}$  and a candidate signature  $\sigma$ , and returns accept or reject;

If  $n = 1$ , we obtain a *designated verifier signature* (DVS) scheme. We say that a signature  $\sigma$  on  $m$  is valid if  $\text{Verify}_{PK_S, PK_D}(m, \sigma) = \text{accept}$ . As usually, we require that an  $n$ -DVS scheme is correct, that is, for all  $(SK_S, PK_S)$  and  $(SK_D, PK_D)$  output by KeyGen, for any  $i \in [1, n]$  and for all  $m \in \mathcal{M}$  we have  $\text{Verify}_{PK_S, PK_D}(\text{Sign}_{SK_S, PK_D}(m)) = \text{Verify}_{PK_S, PK_D}(\text{Simul}_{PK_S, PK_D, SK_{D_i}}(m)) = \text{accept}$ .

Let  $\Delta = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Simul}, \text{Verify})$  be an  $n$ -DVS scheme with the message space  $\mathcal{M}$ . Let  $\Omega$  denote the space from which the random oracle  $H$  is selected;

definition without a random oracle is analogous. Let  $\mathcal{F}_m$  denote the adversary  $\mathcal{F}$  with  $m$  as its input, and we assume that oracle calls are counted as one step.

It is required that a designated verifier signature satisfies the following three properties [LWB05]. We give the definitions of unforgeability and non-transferability only for the sake of completeness since we will not need them in this paper.

*Unforgeability:* Let  $\mathcal{F}$  be an adversary against DVS. We define advantage  $\text{Adv}_{\Delta}^{\text{forge}}(\mathcal{F})$  of  $\mathcal{F}$  to be the next probability:

$$\Pr \left[ \begin{array}{l} H \leftarrow \Omega; (\text{SK}_S, \text{PK}_S) \leftarrow \text{KeyGen}; \\ (\text{SK}_{D_1}, \text{PK}_{D_1}) \leftarrow \text{KeyGen}; \dots; (\text{SK}_{D_n}, \text{PK}_{D_n}) \leftarrow \text{KeyGen}; \\ (m, \sigma) \leftarrow \mathcal{F}^{\text{Sign}_{\text{SK}_S, \text{PK}_D}(\cdot), \text{Simul}_{\text{PK}_S, \text{PK}_D, \text{SK}_{D_i}}(\cdot), H(\cdot)}(\text{PK}_S, \text{PK}_D) : \\ \sigma \notin \Sigma_S(m) \wedge \sigma \notin \Sigma_{D_1}(m) \wedge \dots \wedge \sigma \notin \Sigma_{D_n}(m) \wedge \\ \text{Verify}_{\text{PK}_S, \text{PK}_D}(m, \sigma) = \text{accept} \end{array} \right],$$

where  $\Sigma_S(m)$  is the set of signatures received from  $\text{Sign}_{\text{SK}_S, \text{PK}_D}(m)$  and  $\Sigma_{D_i}(m)$  is the set of signatures received from  $\text{Simul}_{\text{PK}_S, \text{PK}_D, \text{SK}_{D_i}}(m)$ .  $\mathcal{F}$  is said to  $(\tau, q_h, q_s, \varepsilon)$ -forge  $\sigma$  if  $\mathcal{F}$  runs in time at most  $\tau$ , makes at most  $q_h$  hash queries and in total at most  $q_s$  signing and simulation queries, and  $\text{Adv}_{\Delta}^{\text{forge}}(\mathcal{F}) \geq \varepsilon$ . A designated verifier signature scheme is  $(\tau, q_h, q_s, \varepsilon)$ -unforgeable if no forger can  $(\tau, q_h, q_s, \varepsilon)$ -forge it.

*Non-transferability (informal):* given a message-signature pair  $(m, \sigma)$  which is accepted by a designated verifier, and without access to the signer's secret key, it is computationally infeasible to determine whether the message was signed by the signer, or the signature was simulated by the designated verifier.

*Non-delegatability:* Let  $\kappa \in [0, 1]$  be the knowledge error.  $\Delta$  is  $(\tau, \kappa)$ -non-delegatable if there exists a black-box knowledge extractor  $\mathcal{K}$  that, for every algorithm  $\mathcal{F}$  and for every valid signature  $\sigma$ , satisfies the following condition: For every  $(\text{SK}_S, \text{PK}_S) \leftarrow \text{KeyGen}$ ,  $(\text{SK}_{D_i}, \text{PK}_{D_i}) \leftarrow \text{KeyGen}$ , for  $i \in [1, n]$ , and message  $m$ , if  $\mathcal{F}$  produces a valid signature on  $m$  with probability  $\varepsilon > \kappa$ , then on input  $m$  and on access to the oracle  $\mathcal{F}_m$ ,  $\mathcal{K}$  produces one of the secret keys  $(\text{SK}_S, \text{SK}_{D_1}, \dots, \text{SK}_{D_n})$  in expected time  $\frac{\tau}{\varepsilon - \kappa}$  (without counting the time to make the oracle queries). Here,  $\mathcal{F}$ 's probability is taken over the choice of her random coins and over the choice of  $H \leftarrow \Omega$ .

**Variations of  $n$ -DVS.** We call an  $n$ -DVS a *strong  $n$ -DVS* if the verification algorithm also takes an  $\text{SK}_{D_i}$ ,  $i \in [1, n]$ , as an input, and verification without  $\text{SK}_{D_i}$ , for some  $i \in [1, n]$ , is computationally difficult. An  $n$ -DVS scheme is a designated multi verifier signature scheme if verification can be performed only by the coalition of all  $n$  designated verifiers. An  $n$ -DVS scheme is universal if it contains a conventional signing algorithm (w.r.t. no designated verifier) and an arbitrary entity can convert the conventional signature to a signature w.r.t. an arbitrary designated verifier. An  $n$ -DVS scheme is ID-based if the public key of an arbitrary participant  $A$  can be computed from his or her ID  $\text{ID}_A$ .

### 3 Four DVS Schemes and Attacks on Them

#### 3.1 SZM04 Scheme

**Description.** The SZM04 strong ID-based universal DVS scheme [SZM04] can be described as follows:

- Setup: A trusted authority (TA) generates two groups  $(\mathbb{G}, +)$  and  $(\mathbb{H}, \cdot)$  of prime order  $q$  and a bilinear mapping  $\langle \cdot, \cdot \rangle : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{H}$ , together with an arbitrary generator  $P \in \mathbb{G}$ . TA selects a master key  $s \in \mathbb{Z}_q$  and set  $P_{pub} \leftarrow sP$ . Let  $H_{\mathbb{G}} : \{0, 1\}^* \rightarrow \mathbb{G}$  and  $H_q : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  be two random oracles. The system parameters are  $(q, \mathbb{G}, \mathbb{H}, \langle \cdot, \cdot \rangle, P, P_{pub}, H_{\mathbb{G}}, H_q)$ .
- KeyGen( $param$ ):  $S$  and  $D$  publish their identities  $PK_S \leftarrow H_{\mathbb{G}}(ID_S)$  and  $PK_D \leftarrow H_{\mathbb{G}}(ID_D)$ . Their secret keys are defined by TA as  $SK_S \leftarrow s \cdot PK_S$  and  $SK_D \leftarrow s \cdot PK_D$ .
- Sign $_{SK_S, PK_D}(m)$ : to sign a message  $m$  for  $D$ ,  $S$  generates two random values  $k \leftarrow \mathbb{Z}_q, t \leftarrow \mathbb{Z}_q^*$ , and computes  $c \leftarrow \langle PK_D, P \rangle^k, r \leftarrow H_q(m, c), T \leftarrow t^{-1}kP - r \cdot SK_S$ . The signature is  $(T, r, t)$ .
- Simul $_{PK_S, SK_D}(m)$ : To simulate the transcript on an arbitrary message  $m$ ,  $D$  generates random  $R \in \mathbb{G}$  and  $a \in \mathbb{Z}_q^*$ , and computes  $c \leftarrow \langle R, PK_D \rangle \cdot \langle PK_S, SK_D \rangle^a, r \leftarrow H_q(m, c), t \leftarrow r^{-1}a \pmod q, T \leftarrow t^{-1}R$ . The transcript  $(T, r, t)$  is indistinguishable from the real signature [SZM04, Thm. 3].
- Verify $_{PK_S, SK_D}(m, \sigma)$ : given  $(m, T, r, t)$ ,  $D$  verifies its validity by testing whether  $H_q(m, \langle \langle T, PK_D \rangle \cdot \langle PK_S, SK_D \rangle^r \rangle^t) = r$ .

**First Attack.** For both simulation and verification, it is sufficient to know  $\langle SK_S, PK_D \rangle = \langle PK_S, SK_D \rangle$ . Therefore, either the signer or the verifier can delegate the signing rights of  $S$  w.r.t. a fixed designated verifier  $D$ .

**Second Attack.** Assume that the signer discloses  $(k, k \cdot SK_S)$  to any third party  $T$ , where  $k \leftarrow \mathbb{Z}_q^*$ . Given an arbitrary message  $\tilde{m}$  and an arbitrary designated verifier  $D$ ,  $T$  chooses random values  $R \leftarrow \mathbb{G}, a \leftarrow \mathbb{Z}_q^*$  and computes  $\tilde{c} \leftarrow \langle R, PK_D \rangle \cdot \langle k \cdot SK_S, PK_D \rangle^{a(k^{-1}+1)}, \tilde{r} \leftarrow H_q(\tilde{m}, \tilde{c}), \tilde{t} \leftarrow \tilde{r}^{-1}a \pmod q, \tilde{T} \leftarrow \tilde{t}^{-1}R + \tilde{r}k \cdot SK_S$ , obtaining a simulated signature  $(\tilde{T}, \tilde{r}, \tilde{t})$ .  $D$  can verify whether  $H_q(\tilde{m}, \langle \langle \tilde{T}, PK_D \rangle \cdot \langle PK_S, SK_D \rangle^{\tilde{r}} \rangle^{\tilde{t}}) = \tilde{r}$ . The verification accepts since  $\langle \langle \tilde{T}, PK_D \rangle \cdot \langle PK_S, SK_D \rangle^{\tilde{r}} \rangle^{\tilde{t}} = \langle \tilde{t}^{-1}R + \tilde{r}k \cdot SK_S, PK_D \rangle^{\tilde{t}} = \langle R, PK_D \rangle \cdot \langle k \cdot SK_S, PK_D \rangle^{\tilde{r}\tilde{t}} = \langle R, PK_D \rangle \cdot \langle k \cdot SK_S, PK_D \rangle^a$  and

$$\begin{aligned}
 \langle \langle \tilde{T}, PK_D \rangle \cdot \langle PK_S, SK_D \rangle^{\tilde{r}} \rangle^{\tilde{t}} &= \langle \tilde{T}, PK_D \rangle^{\tilde{t}} \cdot \langle PK_S, SK_D \rangle^{\tilde{r}\tilde{t}} \\
 &= \langle R, PK_D \rangle \cdot \langle k \cdot SK_S, PK_D \rangle^a \cdot \langle SK_S, PK_D \rangle^{\tilde{r}\tilde{t}} \\
 &= \langle R, PK_D \rangle \cdot \langle k \cdot SK_S, PK_D \rangle^a \cdot \langle k \cdot SK_S, PK_D \rangle^{ak^{-1}} \\
 &= \langle R, PK_D \rangle \cdot \langle k \cdot SK_S, PK_D \rangle^{a(k^{-1}+1)} = \tilde{c} .
 \end{aligned}$$

Therefore,  $H_q(\tilde{m}, \langle \langle \tilde{T}, PK_D \rangle \cdot \langle PK_S, SK_D \rangle^{\tilde{r}} \rangle^{\tilde{t}}) = H_q(\tilde{m}, \tilde{c}) = \tilde{r}$ . Thus, according to this attack, an arbitrary party who knows  $(k, k \cdot SK_S)$ , for some  $k$ , can simulate signer's signature w.r.t. all designated verifiers.

### 3.2 NSM05 Scheme

The Ng-Susilo-Mu [NSM05] universal designated multi-verifier signature (UDMVS) scheme is as follows (here,  $\mathcal{M} = \mathbb{Z}_q^*$ ):

- Setup: Choose a group pair  $(\mathbb{G}, \mathbb{H})$  of prime order  $|\mathbb{G}| = |\mathbb{H}| = q$ , a bilinear map  $\langle \cdot, \cdot \rangle : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{H}$ , an arbitrary generator  $P \in \mathbb{G}$  and a cryptographic hash function  $H_{\mathbb{G}} : \{0, 1\}^* \rightarrow \mathbb{G}$ . The common parameter is  $param = (q, \mathbb{G}, \mathbb{H}, \langle \cdot, \cdot \rangle, P, H_{\mathbb{G}})$ .
- KeyGen( $param$ ): Given the common parameter, a participant picks a random  $SK \leftarrow \mathbb{Z}_q^*$ , and computes  $PK \leftarrow SK \cdot P$ . The public key is PK and the secret key is SK. Thus,  $S$  has key pair  $(SK_S, PK_S)$  and  $D$  has key pair  $(SK_D, PK_D)$ .
- Sign $_{SK_S, PK_D}(m)$ : Compute  $\hat{\sigma} \leftarrow SK_S \cdot H_{\mathbb{G}}(m)$ ,  $\sigma \leftarrow \langle \hat{\sigma}, \sum_{i=1}^n PK_{D_i} \rangle$ . Return  $\sigma$ .
- Verify $_{PK_S, PK_D, SK_D}(m, \sigma)$ : Each verifier  $D_i$  does the following: compute  $\tilde{\sigma}_i \leftarrow SK_{D_i} \cdot H_{\mathbb{G}}(m)$  and send it to other  $n - 1$  verifiers. After receiving all  $\tilde{\sigma}_j$ ,  $j \neq i$ , validate all  $\tilde{\sigma}_j$  by verifying that  $\langle P, \tilde{\sigma}_j \rangle = \langle PK_j, H_{\mathbb{G}}(m) \rangle$  for  $j \neq i$ ,  $j \in [1, n]$ . Return reject if any of the verifications fails. Return accept if  $\sigma = \prod_{i=1}^n \langle \tilde{\sigma}_i, PK_S \rangle$ , or reject otherwise.

**Attack on NSM05.** Denote  $P_{sum} := \sum_{i=1}^n PK_{D_i}$ . If signer leaks  $SK_S \cdot P_{sum}$  to  $T$ , then  $T$  can compute

$$\sigma \leftarrow \langle H_{\mathbb{G}}(m), SK_S \cdot P_{sum} \rangle = \langle SK_S \cdot H_{\mathbb{G}}(m), P_{sum} \rangle = \langle \hat{\sigma}, P_{sum} \rangle .$$

After receiving  $(m, \sigma)$ , each verifier  $i$  computes  $\tilde{\sigma}_i \leftarrow SK_{D_i} \cdot H_{\mathbb{G}}(m)$ , and verifies that  $\langle P, \tilde{\sigma}_j \rangle = \langle PK_j, H_{\mathbb{G}}(m) \rangle$  for  $j \neq i$ ,  $j \in [1, n]$ . Now,  $\sigma = \prod_{i=1}^n \langle \tilde{\sigma}_i, PK_S \rangle$  since

$$\begin{aligned} \sigma &= \langle H_{\mathbb{G}}(m), SK_S \cdot P_{sum} \rangle = \langle SK_S \cdot H_{\mathbb{G}}(m), P_{sum} \rangle = \langle \hat{\sigma}, P_{sum} \rangle \\ &= \prod_{i=1}^n \langle \hat{\sigma}, SK_{D_i} \cdot P \rangle = \prod_{i=1}^n \langle SK_S \cdot H_{\mathbb{G}}(m), SK_{D_i} \cdot P \rangle \\ &= \prod_{i=1}^n \langle SK_{D_i} \cdot H_{\mathbb{G}}(m), SK_S \cdot P \rangle \\ &= \prod_{i=1}^n \langle \tilde{\sigma}_i, PK_S \rangle . \end{aligned}$$

Therefore, for any message  $m$ ,  $T$  can simulate signer's signature  $\sigma$  and pass the verification equation. Note that alternatively, all verifiers can cooperate by leaking  $\sum SK_{D_i} \cdot PK_S = SK_S \cdot P_{sum}$ . Therefore, the NSM05 scheme is delegatable.

Additionally, Ng, Susilo and Mu first proposed a “simple” UDMVS scheme that is based on the universal DVS from [SBWP03]. There, analogously, if signer leaks  $SK_S \cdot PK_{D_i}$  to  $T$ , then  $T$  can compute  $\sigma_i = \langle SK_S \cdot PK_{D_i}, H_{\mathbb{G}}(m) \rangle = \langle PK_{D_i}, \hat{\sigma} \rangle$ , for  $i \in [1, n]$ . Verifier will accept  $\sigma_i$  for that  $\sigma_i = \langle SK_S \cdot PK_{D_i}, H_{\mathbb{G}}(m) \rangle = \langle SK_S SK_{D_i} P, H_{\mathbb{G}}(m) \rangle = \langle SK_S \cdot P, H_{\mathbb{G}}(m) \rangle^{SK_{D_i}} = \langle PK_S, H_{\mathbb{G}}(m) \rangle^{SK_{D_i}}$  for  $i \in [1, n]$ .

Furthermore, our attack works also with the MDVS scheme from [NSM05] because its signing algorithm is same as the signing algorithm in the UDMVS scheme.

### 3.3 ZFI05 Scheme

The next strong DVS scheme ZFI05 was proposed in [ZFI05] (we describe a slightly simplified version of ZFI05, but our attack works also with the original version):

- Setup: Choose a bilinear group pair  $(\mathbb{G}, \mathbb{H})$  of prime order  $|\mathbb{G}| = |\mathbb{H}| = q$ , with a bilinear map  $\langle \cdot, \cdot \rangle : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{H}$  and an isomorphism  $\psi : \mathbb{H} \rightarrow \mathbb{G}$ . Here,  $\mathbb{G}$  is multiplicative. Choose a random generator  $g_2 \in \mathbb{H}$ , and compute  $g_1 = \psi(g_2) \in \mathbb{G}$ . Then the common parameter is  $param = (q, \mathbb{G}, \mathbb{H}, \langle \cdot, \cdot \rangle, \psi, g_1, g_2)$ .
- KeyGen( $param$ ): Pick random  $x, y \leftarrow \mathbb{Z}_q^*$ , compute  $u \leftarrow g_2^x, v \leftarrow g_2^y$ . The public key is  $PK \leftarrow (u, v)$  and the secret key is  $SK \leftarrow (x, y)$ . In particular,  $S$  has a key pair with  $PK_S = (u_S, v_S)$ ,  $SK_S = (x_S, y_S)$  and  $D$  has a key pair with  $PK_D = (u_D, v_D)$ ,  $SK_D = (x_D, y_D)$ .
- Sign $_{SK_S, PK_D}(m)$ : Pick a random  $r \leftarrow \mathbb{Z}_q^*$ . If  $x_S + r + y_S m \equiv 0 \pmod q$ , restart. Compute  $\sigma' \leftarrow g_1^{1/(x_S+r+y_S m)} \in \mathbb{G}$ ,  $h \leftarrow g_2^r, d \leftarrow \langle u_D, v_D^r \rangle \in \mathbb{H}$ . Return  $\sigma \leftarrow (\sigma', h, d)$ .
- Simul $_{PK_S, SK_D}(m)$ : Pick a random  $s \in \mathbb{Z}_q^*$  and compute  $\sigma' \leftarrow g_2^s, h \leftarrow g_2^{1/s} u_S^{-1} v_S^{-m}$  and  $d \leftarrow \langle g_1, h \rangle^{x_D y_D}$ . Return  $\sigma \leftarrow (\sigma', h, d)$ .
- Verify $_{PK_S, SK_D}(\sigma', h, d)$ : Output accept if  $\langle g_1, g_2 \rangle = \langle \sigma', u_S \cdot h \cdot v_S^m \rangle$  and  $d = \langle u_D, h^{y_D} \rangle$ . Otherwise, output reject.

**Attack on ZFI05.** In simulation algorithm, the designated verifier can compute  $d$  as  $d \leftarrow \langle g_1^{x_D y_D}, h \rangle$ . Thus, designated verifier can reveal  $g_1^{x_D y_D}$ , and therefore this scheme is delegatable by the verifier. Note that the delegation token does not depend on the signer.

### 3.4 LV04 Scheme

**Description.** In [LV04b], Laguillaumie and Vergnaud proposed the next 2-DVS scheme based on bilinear maps. Here,  $D_1$  and  $D_2$  are two verifiers specified by signer  $S$ .  $\mathbb{G}$  and  $\mathbb{H}$  are groups of order  $q$ ,  $P$  generates  $\mathbb{G}$ , and  $\langle \cdot, \cdot \rangle : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{H}$  is an admissible bilinear map. Let  $BDHGen$  be a Bilinear Diffie-Hellman (BDH)-prime order generator [LV04b].

- Setup: Set  $(q, \mathbb{G}, \mathbb{H}, \langle \cdot, \cdot \rangle, P) \leftarrow BDHGen$ , and let  $H_{\mathbb{G}}$  be a random member of a hash function family from  $\{0, 1\}^* \times \mathbb{H} \rightarrow \mathbb{G}$ . The common parameter is  $(q, \mathbb{G}, \mathbb{H}, \langle \cdot, \cdot \rangle, P, H_{\mathbb{G}})$ .
- KeyGen( $param$ ): Pick a random  $SK \leftarrow \mathbb{Z}_q^*$ , and compute  $PK \leftarrow SK \cdot P$ . The public key is  $PK$  and the secret key is  $SK$ .
- Sign $_{SK_S, PK_{D_1}, PK_{D_2}}(m)$ : Given a message  $m \in \{0, 1\}^*$ ,  $S$  picks at random two integers  $(r, \ell) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ , computes  $u \leftarrow \langle PK_{D_1}, PK_{D_2} \rangle^{SK_S}$ ,  $Q_1 \leftarrow SK_S^{-1}(H_{\mathbb{G}}(m, u^\ell) - r(PK_{D_1} + PK_{D_2}))$  and  $Q_2 \leftarrow rP$ . The signature is  $\sigma = (Q_1, Q_2, \ell)$ .
- Verify $_{PK_S, PK_D, SK_{D_i}}(m, Q_1, Q_2, \ell)$ : Designated verifier  $D_i$ , where  $i \in \{1, 2\}$ , computes  $u \leftarrow \langle PK_S, PK_{D_{3-i}} \rangle^{SK_{D_i}}$ . He or she tests whether  $\langle Q_1, PK_S \rangle \cdot \langle Q_2, PK_{D_1} + PK_{D_2} \rangle = \langle H_{\mathbb{G}}(m, u^\ell), P \rangle$ .



**Attack on LV04.** Suppose  $D_1$  and  $D_2$  collude to leak  $\text{SK}_{D_1} + \text{SK}_{D_2}$  to  $T$ . Then  $T$  picks two random integers  $\tilde{r}, \tilde{\ell} \leftarrow \mathbb{Z}_q^*$ , computes  $\tilde{M} \leftarrow H_{\mathbb{G}}(m, \tilde{\ell})$ ,  $\tilde{Q}_1 \leftarrow \tilde{r}P$ , and  $\tilde{Q}_2 \leftarrow (\text{SK}_{D_1} + \text{SK}_{D_2})^{-1}(\tilde{M} - \tilde{r} \cdot \text{PK}_S)$ . The simulated signature is  $\tilde{\sigma} \leftarrow (\tilde{Q}_1, \tilde{Q}_2, \tilde{\ell})$ . Verification accepts since

$$\begin{aligned} & \langle \tilde{Q}_1, \text{PK}_S \rangle \cdot \langle \tilde{Q}_2, \text{PK}_{D_1} + \text{PK}_{D_2} \rangle \\ &= \langle \tilde{r}P, \text{PK}_S \rangle \cdot \langle (\text{SK}_{D_1} + \text{SK}_{D_2})^{-1}(\tilde{M} - \tilde{r} \cdot \text{PK}_S), \text{SK}_{D_1}P + \text{SK}_{D_2} \cdot P \rangle \\ &= \langle \tilde{r}P, \text{PK}_S \rangle \cdot \langle (\text{SK}_{D_1} + \text{SK}_{D_2})^{-1}(\tilde{M} - \tilde{r} \cdot \text{PK}_S), P \rangle^{\text{SK}_{D_1} + \text{SK}_{D_2}} \\ &= \langle \tilde{r}P, \text{PK}_S \rangle \cdot \langle \tilde{M} - \tilde{r} \cdot \text{PK}_S, P \rangle \\ &= \langle \tilde{M}, P \rangle \cdot \langle \tilde{r} \cdot \text{PK}_S, P \rangle \cdot \langle -\tilde{r} \cdot \text{PK}_S, P \rangle = \langle \tilde{M}, P \rangle . \end{aligned}$$

Therefore, if  $D_1$  and  $D_2$  collaborate then they can leak  $\text{SK}_{D_1} + \text{SK}_{D_2}$  to  $T$ . After that,  $T$  will be able to simulate signatures of *any* signer w.r.t. the designated verifier pair  $(D_1, D_2)$ .

**Discussion.** Our attack is based on the following observation: by the verification equation  $\langle Q_1, \text{PK}_S \rangle \cdot \langle Q_2, \text{PK}_{D_1} + \text{PK}_{D_2} \rangle = \langle H_{\mathbb{G}}(m, u^\ell), P \rangle$ , we have  $\langle Q_1, P \rangle^{\text{SK}_S} \cdot \langle Q_2, P \rangle^{\text{SK}_{D_1} + \text{SK}_{D_2}} = \langle H_{\mathbb{G}}(m, u^\ell), P \rangle$ . Here, attacker can choose only  $Q_1$  and  $Q_2$ , and it must hold that  $\text{SK}_S \cdot Q_1 + (\text{SK}_{D_1} + \text{SK}_{D_2})Q_2 = H_{\mathbb{G}}(m, u^\ell)$ . Due to the random oracle assumption,  $H_{\mathbb{G}}(m, u^\ell)$  is a random value, and thus either  $Q_1$  or  $Q_2$  must depend on  $H_{\mathbb{G}}(m, u^\ell)$ . This means that attacker either must know the value  $\text{SK}_S$  (and thus he can recover  $S$ 's secret key), or the value  $\text{SK}_{D_1} + \text{SK}_{D_2}$ . Leaking  $\text{SK}_S$  is not an attack according to [LWB05], but leaking  $\text{SK}_{D_1} + \text{SK}_{D_2}$  is.

To guarantee the non-transferability, the designated verifier(s) should have the capability to simulate correct signature transcripts. However, the LV04 scheme does not include simulation algorithms. The above attack can also be treated as two-party simulation algorithm if  $D_1$  and  $D_2$  execute it themselves.

Although a third party cannot deduce  $\text{SK}_{D_1}$  or  $\text{SK}_{D_2}$  from  $\text{SK}_{D_1} + \text{SK}_{D_2}$ , we need that two parties  $D_1$  and  $D_2$  compute  $\text{SK}_{D_1} + \text{SK}_{D_2}$  together. This means that either these two parties must trust each other, or they have to execute a secure two-party computation.

Finally, note that both this attack and the attack against ZFI05 have the same feature: if the delegation token revealed by verifier(s) (either  $\text{SK}_{D_1} + \text{SK}_{D_2}$  or  $g_1^{x_{D_1}y_{D_1}}$ ) is not connected anyhow to the signer. Therefore, after revealing those values, a third party can simulate the signature of *any* signer w.r.t. a fixed designated verifier or a fixed pair of designated verifiers.

## 4 On Different Delegation Attacks

**Who can delegate?** In the previous section, we saw at least two kinds of delegation attacks:

**Attack I:** Either the signer or one of the designated verifiers can delegate the signing rights to a third party  $T$  without disclosing his or her secret key.

**Attack II:** One of the designated verifiers (or even only the coalition of all verifiers) can delegate the signing right to a third party without disclosing his or her secret key, while the signer cannot do it.

The non-delegatability notion introduced in [LWB05] corresponds to security against Attack I. Next, we will give a somewhat formal definition of what we mean by a vulnerability to Attack II. (Intuitively, it says that a  $n$ -DVS scheme  $\Delta$  is verifier-only delegatable if it is delegatable but it cannot be delegated by the signer without leaking Signer's secret key.)

*Verifier-only delegatability:* As previously,  $\mathcal{F}_m$  denotes  $\mathcal{F}$  with  $m$  as its input, and oracle calls are counted as one step. More precisely, let  $\kappa \in [0, 1]$  be the knowledge error. We say that  $\Delta$  is verifier-only  $(\tau, \kappa)$ -delegatable if it is not  $(\tau, \kappa)$ -non-delegatable and there exists a black-box knowledge extractor  $\mathcal{K}$  that, for every algorithm  $\mathcal{F}$  and for every message  $m \in \mathcal{M}$  satisfies the following condition: for every  $(SK_S, PK_S) \leftarrow \text{KeyGen}$ ,  $(SK_{D_1}, PK_{D_1}) \leftarrow \text{KeyGen}$ ,  $\dots$ ,  $(SK_{D_n}, PK_{D_n}) \leftarrow \text{KeyGen}$ , for every bit-string  $d$  (delegation token) that does not depend on  $SK_{D_i}$  for any  $i \in [1, n]$ , and for any message  $m$ , if  $\mathcal{F}$  produces a valid signature on  $m$  with probability  $\varepsilon > \kappa$  then, on input  $m$  and on access to the oracle  $\mathcal{F}_m$ ,  $\mathcal{K}$  produces  $SK_{D_i}$  for some  $i \in [1, n]$  in expected time  $\frac{\tau}{\varepsilon - \kappa}$  (without counting the time to make the oracle queries). Here again,  $\mathcal{F}$ 's probability is taken over the choice of her random coins and over the choice of the random oracles.

**What exactly can be delegated?** The presented attacks can be divided into delegation attacks that allow to delegate the signing rights of a fixed signer w.r.t. a fixed tuple of designated verifiers, or the rights of *any* signer w.r.t. a fixed tuple of designated verifiers, or the rights of a fixed signer w.r.t. any tuple of designated verifiers. According to [LWB05], existence of any of these attacks makes a scheme delegatable. Again, it can be argued that the first type of attack is the most serious one since the last two attack types give too much power to the third party  $T$  (and therefore one might be less motivated to delegate the rights). One can try to modify the delegatability definition so that only the first attack type is classified as an attack. We will leave it as an open question whether this is reasonable.

**Final Remarks.** Regardless of the previous comments, our own opinion is that our attacks against all four schemes indicate some weaknesses in them and while the non-delegatability definition of [LWB05] might be too strong in some sense, to avoid any kind of future attack and unexpected vulnerabilities, it is a good idea to design DVS schemes that are non-delegatable according to [LWB05].

## Acknowledgements

The first and the third authors are partially supported by the National Grand Fundamental Research 973 Program of China under Grant No. G1999035804. The second author is partially supported by the Estonian Science Foundation, grant 6096. The authors would like to thank Guilin Wang and the anonymous referees for useful comments.

## References

- [Cha96] David Chaum. Private Signature and Proof Systems, 1996. US-patent no 5,493,614.
- [JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154, Saragossa, Spain, May 12–16, 1996. Springer-Verlag.
- [LV04a] Fabien Laguillaumie and Damien Vergnaud. Designated Verifier Signatures: Anonymity and Efficient Construction from Any Bilinear Map. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks, 4th International Conference, SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 105–119, Amalfi, Italy, September 8–10, 2004. Springer Verlag.
- [LV04b] Fabien Laguillaumie and Damien Vergnaud. Multi-designated Verifiers Signatures. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *Information and Communications Security, 6th International Conference, ICICS 2004*, volume 3269 of *Lecture Notes in Computer Science*, pages 495–507, Malaga, Spain, October 27–29, 2004. Springer-Verlag.
- [LWB05] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated Verifier Signature Schemes: Attacks, New Security Notions and A New Construction. In Luis Caires, Giuseppe F. Italiano, Luis Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 459–471, Lisboa, Portugal, 2005. Springer-Verlag.
- [NSM05] Ching Yu Ng, Willy Susilo, and Yi Mu. Universal Designated Multi Verifier Signature Schemes. In Cheng-Zhong Xu and Laurence T. Yang, editors, *The International Workshop on Security in Networks and Distributed Systems (SNDS 2005)*, Fukuoka, Japan, July 20–22, 2005. IEEE Press. To appear.
- [SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal Designated-Verifier Signatures. In Chi Sung Lai, editor, *Advances on Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542, Taipei, Taiwan, November 30–December 4, 2003. Springer-Verlag.
- [SKM03] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An Efficient Strong Designated Verifier Signature Scheme. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003*, volume 2971 of *Lecture Notes in Computer Science*, pages 40–54, Seoul, Korea, November 27–28, 2003. Springer-Verlag.
- [SWP04] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 86–100, Singapore, March 1–4, 2004. Springer-Verlag.
- [SZM04] Willy Susilo, Fangguo Zhang, and Yi Mu. Identity-Based Strong Designated Verifier Signature Schemes. In Josef Pieprzyk and Huaxiong Wang, editors, *The 9th Australasian Conference on Information Security and Privacy (ACISP 2004)*, volume 3108 of *Lecture Notes in Computer Science*, pages 313–324, Sydney, Australia, July 13–15, 2004. Springer-Verlag.

- [ZFI05] Rui Zhang, Jun Furukawa, and Hideki Imai. Short Signature and Universal Designated Verifier Signature Without Random Oracles. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 483–498, New York, NY, USA, June 7–10, 2005. Springer-Verlag.

# PIATS: A Partially Sanitizable Signature Scheme

Tetsuya Izu, Nobuyuki Kanaya, Masahiko Takenaka, and Takashi Yoshioka

Fujitsu Laboratories Ltd.,  
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan  
{izu, kanaya, takenaka, yoshioka}@labs.fujitsu.com

**Abstract.** In e-government or e-tax payment systems, appropriate alterations on digitally signed documents are required to hide personal information, namely privacy. Standard digital signature schemes do not allow such alterations on the signed documents since there is no means to distinguish appropriate alterations from inappropriate forgeries. The *sanitizable signature scheme* is a possible solution for such systems in which sanitizings of partial information are possible, after a signature is signed on the original (unsanitized) document. However, in previously proposed schemes, since sanitizers are anonymous, verifiers cannot identify sanitizers, and thus dishonest sanitizings are possible. This paper proposes a new sanitizable signature scheme “PIATS” in which partial information can be sanitized. Moreover, verifiers can identify sanitizers and thus dishonest sanitizings are eliminated.

**Keywords:** Sanitizable signature scheme, partial integrity, privacy.

## 1 Introduction

To governmental, municipal or military offices, there is a strong demand to disclose documents they hold or held. In fact, many countries have disclosure laws for these organizations. However, such disclosed documents should exclude personal or national secret information because of privacy or diplomatic reasons. In old days, paper documents were blacked-out by physical maskings to hide information. Unfortunately, we have no analogous system for electronic documents. For example, the New York Times website exposed CIA agents with carelessness, since they sanitized the electronic document by hand [Wir02]. Other example is an exposure of the Carnivore review team by the Justice Department of USA [Wir00]. Thus a systematic sanitizing method for electronic documents are required in this internet era. On the other hand, signatures are very common technology to assure the integrity of the document, since it detects inappropriate forgeries on original documents. However, current signature schemes can not distinguish appropriate alternations, namely sanitizations as mentioned above, and adversaries' inappropriate forgeries. Thus we require a new document managing system which establish the privacy of some part of documents (by sanitizations) and the integrity of other parts (by enhanced signature technology).

The *sanitizable signature scheme* (or the *content extractable signature scheme*) is a possible solution in which partial information are sanitizable even after a signature is signed on the original document [SBZ01, MSI+03, MIM+05, ACM+05]. In these sanitizable signature schemes, appropriate alternations (sanitizings) and inappropriate alternations (forgeries) are distinctly treated, namely, all sanitizations are allowed but any forgeries are not. Thus these schemes guarantee the integrity of the original document with hiding the privacy. However, previously proposed sanitizable signature schemes required rather severe limitations or had serious security problems. In SUMI-1, a sanitization is allowed only once [MSI+03]. In CES schemes and SUMI-2,3,4, multiple sanitizations are allowed [SBZ01, MSI+03], however, since sanitizers are anonymous, verifiers cannot identify sanitizers. More worse, dishonest sanitizations are possible in SUMI-4 [MIM+05]. In a recent scheme SUMI-5 [MIM+05], such dishonestly sanitizations are avoided, but limitations are somewhat strengthened. In fact, SUMI-5 assumes a situation where sanitizers can control the disclosing criteria of the following sanitizers.

This paper proposes a new partially sanitizable signature scheme “PIATS” (Partial Information Assuring Technology for Signature) in which dishonestly sanitizations on any part of closed information are detected. Moreover, for sanitized information, verifiers can identify not only which part is sanitized but also who sanitized from a signer and sanitizers. Since any provably secure digital signature schemes such as RSA-PSS [PKCS] can be combined with the proposed scheme, we can establish an electronic document management system which assures the integrity of the document with hiding personal information.

The rest of the paper is organized as follows: section 2 describes previously proposed sanitizable signature schemes CES families and SUMI families. Then, section 3 proposes our sanitizable signature scheme with some discussions. A comparison of mentioned schemes are in section 4.

## 2 Preliminaries

In this section, we briefly introduce previously proposed sanitizable signature schemes CES families [SBZ01] and SUMI families [MSI+03, MIM+05].

### 2.1 Notations

In this paper, we assume that an original document to be digitally signed is given as an  $n$ -block data  $\{m_i\}_{1 \leq i \leq n}$ . Here length of each block can be distinct. For example,  $m_i$  can be minimal components in XML document. A value  $r$  is a (pseudo) random value generated by an appropriate generator. A function  $\text{Hash}(\cdot)$  is an arbitrary secure hash function such as SHA-256<sup>1</sup>. Functions  $\text{Sign}(\cdot)/\text{Verify}(\cdot)$  are signing/verifying functions of a non-sanitizable provably secure signature scheme such as RSA-PSS [PKCS].

---

<sup>1</sup> As in [SBZ01], all hash functions in this paper can be replaced by preimage resistant and 2nd preimage resistant functions. However, for simplicity, we denote just “hash functions”.

## 2.2 Three-Party Model

In the followings, we use a *three-party model* of sanitizable signature schemes as in previous schemes [MSI+03, MIM+05, SBZ01]. In this model, three parties, *signers*, *sanitizers* and *verifiers* are considered as players <sup>2</sup>.

**Signer** assures the integrity of an original document by digitally signing a signature. The signer does not know which part of the document will be sanitized in future when they sign.

**Sanitizers** determine which part of the document to be sanitized and actually sanitize the document, on input a signed document and a signature from the signer (and a sanitized document if the sanitizer is not the first sanitizer). Here, we assume that sanitizers cannot create a new document, and sanitizers cannot change the original document nor signature.

**Verifiers** confirm the integrity of the document by verifying the original signature, and confirm whether it was signed by an appropriate signer and was sanitized by appropriate sanitizers.

## 2.3 CES

CES (Content Extracting Signature) is a family of sanitizable signature schemes proposed by Steinfeld-Bull-Zheng [SBZ01]. CES family has four schemes CES-CV, CES-HT, CES-RSAP, and CES-MERP. CES-CV is a main scheme and remaining are its variants. CES-CV and CES-HT can be combined with arbitrary signature schemes, while CES-RSAP and CES-MERP can be combined with only RSA-type signature schemes. Since CES-CV is very similar to SUMI-4 in the following section, we do not introduce CES schemes in detail here.

## 2.4 SUMI

SUMI is a family of sanitizable signature schemes proposed by Miyazaki et al. successively [MSI+03, MIM+05]. SUMI family has five schemes SUMI-1, SUMI-2, SUMI-3, SUMI-4, and SUMI-5 <sup>3</sup>. All of these schemes can be combined with arbitrary signature schemes.

**SUMI-1, SUMI-2, SUMI-3:** On an original  $n$ -block document, SUMI-1 generates signatures for all possible subsets of the document (namely, a signer generates  $2^n$  signatures). A sanitizer determines disclosing blocks and publishes a corresponding subset and a signature. Thus a sanitization is allowed only once in SUMI-1. In addition, SUMI-1 is far from efficient since it requires  $2^n$  signatures, which is exponential to the size of the original document.

SUMI-2 generates  $n$  signatures corresponding to  $n$ -blocks. A sanitizer determines disclosing blocks and publishes a corresponding index set and signatures.

<sup>2</sup> In [SBZ01], sanitizers are described as *owners*.

<sup>3</sup> “Sumi (Indian ink)” is a standard writing material in eastern Asian countries including Japan, and is used for non-digital sanitizations.

**Table 1.** A description of SUMI-4**Signer**

- 
1. For a given original document  $\{m_i\}_{1 \leq i \leq n}$ , a signer pads random values  $r_i$  to each block.
  2. For a padded document  $M = \{(m_i, r_i)\}_{1 \leq i \leq n}$ , generate a set of hash values  $H = \{h_i = \text{Hash}(m_i, r_i)\}_{1 \leq i \leq n}$  for a given hash function  $\text{Hash}(\cdot)$ .
  3. Generate a signature  $s = \text{Sign}_{\text{signer}}(H)$  for a given signer's signing function  $\text{Sign}_{\text{signer}}(\cdot)$ .
  4. Output  $(M, s)$  as an original document and a signature.
- 

**Sanitizer**

- 
1. Determine a disclosing index set  $D \subset \{1, \dots, n\}$ , where blocks  $(m_i, r_i)$  ( $i \in D$ ) will be disclosed.
  2. A sanitizer converts the document  $M$  to a new document  $\tilde{M} = \{\tilde{m}_i\}_{1 \leq i \leq n}$ , where

$$\tilde{m}_i = \begin{cases} (m_i, r_i) & \text{if } i \in D \\ h_i & \text{if } i \notin D. \end{cases}$$

3. Output the sanitized document  $\tilde{M}$  and the index set  $D$ .
- 

**Verifier**

- 
1. On input an original signature  $s$ , a sanitized document  $\tilde{M} = \{\tilde{m}_i\}_{1 \leq i \leq n}$ , and a disclosing index set  $D$ , generate a set of hash values  $H' = \{h'_i\}_{1 \leq i \leq n}$ , where

$$h'_i = \begin{cases} \text{Hash}(\tilde{m}_i) & \text{if } i \in D \\ \tilde{m}_i & \text{if } i \notin D. \end{cases}$$

2. Compute  $\text{Verify}_{\text{signer}}(H', s)$  and confirm the integrity of disclosed parts of the document for a signer's verifying function  $\text{Verify}_{\text{signer}}(\cdot)$ .
- 

Thus multiple sanitizations are possible in SUMI-2. But SUMI-2 is not efficient since it requires  $n$  signatures.

SUMI-3 generates  $n$  hash values (instead of signatures) corresponding to  $n$ -blocks and a signature on a concatenation of these hash values. A sanitizer determines disclosing blocks and publishes a corresponding index set and the updated document which consists of original blocks for disclosing parts and hash values for closing parts. Thus multiple sanitizations are possible in SUMI-3. Moreover, SUMI-3 is efficient. However, it is not secure because the procedure is deterministic [MSI+03]. This idea is inherited to the following SUMI-4.

**SUMI-4:** In SUMI-4, in the beginning, all blocks are padded by random values to enhance the security. Then a set of hash values of all padded blocks and a signature on a concatenation of these hash values are generated. When a sanitizer sanitizes a specified block, he/she replaces the block to the corresponding hash



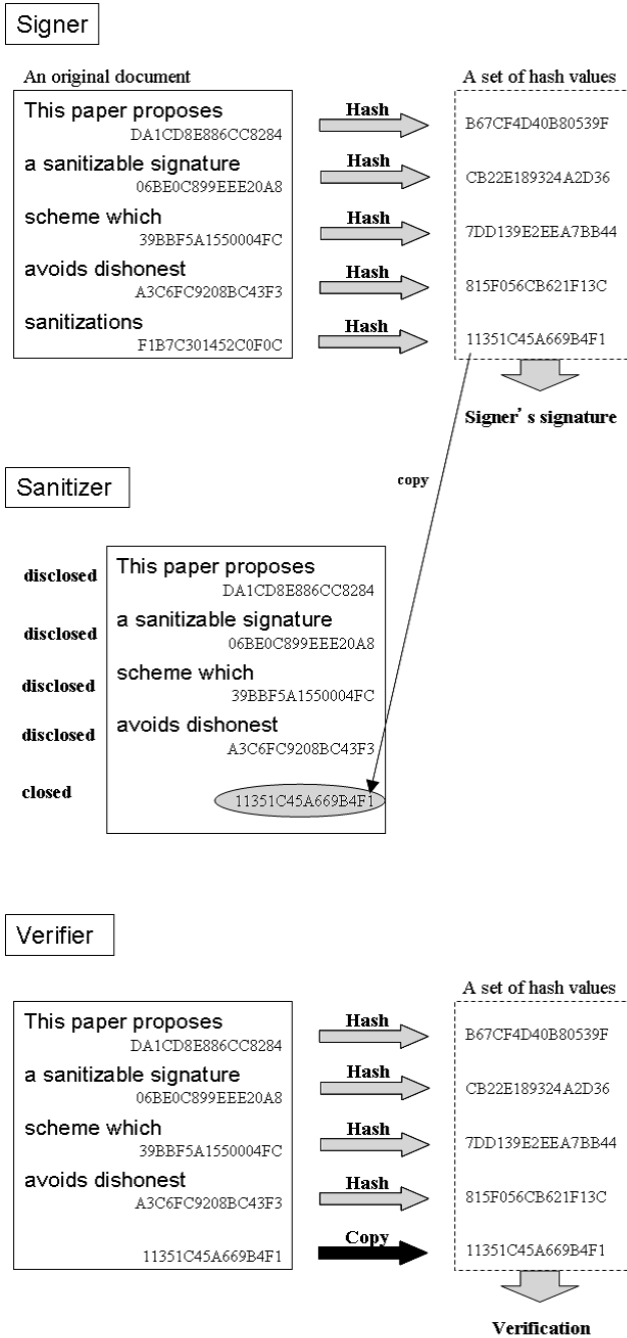


Fig. 1. An outline of SUMI-4

value. Thus multiple sanitizations are possible in SUMI-4. A formal description of SUMI-4 is in Table 1 (see also Figure 1).

Similar to SUMI-3, SUMI-4 is an efficient sanitizable signature scheme since it only requires 1 signature. Moreover, multiple sanitizations are possible in SUMI-4. However, SUMI-4 has the following security problem.

**Additional Sanitization Attack:** In all of SUMI-2, SUMI-3, SUMI-4, and CES schemes, multiple sanitizations are possible. In [MIM+05], Miyazaki et al. observed that this property allows an adversary to the *additional sanitization attack*, a variant of the man-in-the-middle attack, in which the adversary intercepts an appropriately sanitized document, and sends the corruptly sanitized document to a verifier to which dishonestly sanitizations are added. This is because sanitizers are anonymous in these schemes; in order to resist this kind of attacks, anonymous sanitizations should be avoided.

**SUMI-5:** Miyazaki et al. proposed an enhanced sanitizable signature scheme SUMI-5 in the same paper [MIM+05]. Since SUMI-5 is based on SUMI-4, SUMI-5 is also very efficient and multiple sanitizations are possible. Moreover, adversaries' dishonestly sanitizations are avoided. A formal description of SUMI-5 is in Table 2.

In SUMI-5, dishonest sanitizations are avoided by using three sets  $D_n$ ,  $D_{na}$ ,  $C$ . By changing these sets, multiple sanitizations are possible, however, since each sets should be monotonously increasing or decreasing, latter sanitizers can prohibit sanitizations beyond the disclosing conditions determined by previous sanitizers. On the other hand, because the signer cannot determine any conditions on the sanitization, the first sanitizer cannot determine the disclosing condition.

SUMI-5 is a very interesting scheme because it firstly excludes dishonest sanitizations. However, since a closing condition should be determined in the beginning, it seems impractical. Our study is motivated by this problem. Similar to SUMI-5, our proposed scheme "PIATS" is based on SUMI-4 rather than SUMI-5, but proceeds another direction as in the next section.

### 3 Proposed Scheme

In this section, we propose a new partially sanitizable signature scheme "PIATS" (Partial Information Assuring Technology for Signature) which supports multiple sanitizations with avoiding dishonest sanitizations. In order to resist the additional sanitization attack and solve the problem of SUMI-5 (discussed in the previous section), we establish three conditions which the sanitizable signature scheme should satisfy:

- (C1) Signers cannot determine disclosed blocks.
- (C2) Sanitizers and their sanitized parts can be identified by verifiers.
- (C3) Sanitizers cannot control other blocks than their sanitizing blocks.

Note that SUMI-5 satisfies the condition (C1) and (C2), but not (C3). This may be a main reason why SUMI-5 requires those severe limitations on index sets.

**Table 2.** A description of SUMI-5**Signer**

- 
1. For a given original document  $\{m_i\}_{1 \leq i \leq n}$ , a signer pads random values  $r_i$  to each block. Also generate random values  $s_i$  for each block. Let  $M = \{(m_i, r_i)\}_{1 \leq i \leq n}$  be a padded document and  $S = \{s_i\}_{1 \leq i \leq n}$  be a set of random values.
  2. For an  $i$ -th block, draw a line  $\ell_i$  passing two points  $(1, \text{Hash}(m_i, r_i))$  and  $(2, \text{Hash}(s_i))$  for a given hash function  $\text{Hash}(\cdot)$ . Next, compute two values  $P_i, Q_i$  such that two points  $(0, Q_i), (3, P_i)$  are on the line  $\ell_i$ . Let  $P = \{P_i\}_{1 \leq i \leq n}$  be a set of  $P_i$  values.
  3. Generate a signature  $s = \text{Sign}_{\text{signer}}(Q_1 || \dots || Q_n || P_1 || \dots || P_n)$  for a signer's signing function  $\text{Sign}_{\text{signer}}(\cdot)$ , where  $||$  denotes a concatenation.
  4. Output  $(M, S, P, s)$  as an original document and a signature.
- 

**Sanitizer**

- 
1. Determine three index sets (partitions)  $D_a, D_{na}, C \subset \{1, \dots, n\}$  such that  $D_a \cup D_{na} \cup C = \{1, \dots, n\}$  and  $D_a \cap D_{na} = D_{na} \cap C = C \cap D_a = \phi$ . For an index  $i \in D_a$ , the  $i$ -th block is disclosed and additional sanitization is allowed. For an index  $i \in D_{na}$ , the  $i$ -th block is disclosed but additional sanitization is not allowed. On the other hand, for an index  $i \in C$ , the  $i$ -th block is closed at all times.
  2. Let  $\tilde{M} = M \setminus \{(m_i, r_i)\}_{i \in C}$ ,  $\tilde{S} = S \setminus \{s_i\}_{i \in D_{na}}$ .
  3. Output the sanitized document  $(\tilde{M}, \tilde{S})$  with the index sets  $D_n, D_{na}, C$ .
- 

**Verifier**

- 
1. On input an original signature  $s$ , a sanitized document  $\tilde{M} = \{\tilde{m}_i\}$  with sets  $\tilde{S}, P$ , and an index set  $C$ , for each block, draw a line  $\ell'_i$  passing two points  $(1, \text{Hash}(\tilde{m}_i))$  and  $(3, P_i)$  if this block is disclosed or  $(2, \text{Hash}(s_i))$  and  $(3, P_i)$  if this block is closed.
  2. For each line  $\ell'_i$ , compute  $Q'_i$  such that a point  $(0, Q'_i)$  is on the line  $\ell'_i$ .
  3. Compute  $\text{Verify}_{\text{signer}}(Q'_1 || \dots || Q'_n || P_1 || \dots || P_n, s)$  and confirm the integrity of disclosed parts of the document for a signer's verifying function  $\text{Verify}_{\text{signer}}(\cdot)$ .
- 

**3.1 Approach**

After analyzing the previous sanitizable signature schemes, we reached a conclusion that a main reason why dishonest sanitizations are possible is because the anonymity of the sanitizers, namely there are no identifications in sanitizations. We consider that identifications of a signer and sanitizers are the most required property for secure sanitizable signature schemes.

In order to establish a new scheme, we went back to SUMI-4 rather than SUMI-5. In SUMI-4, a signature scheme assures the integrity of a hash set  $H$ , and some properties of the hash function (the preimage resistance and the 2nd

preimage resistance) assures the integrity of the sanitized document. In other words, previous sanitizable signature schemes generate a sign on a hash value set  $H$  and verify the integrity of  $H$ , while standard (non-sanitizable) signature schemes generate a sign on a document  $M$  and verify the integrity of  $M$ . With this property, sanitizable schemes can verify the integrity of the sanitized document by verifying the integrity of  $H$  without a direct access on the document.

### 3.2 Description of the Proposed Scheme

This section describes a proposed sanitizable signature scheme “PIATS”, in which a signature on an original document is generated from a hash value set  $H$  and its signature. Closing blocks and corresponding padded random values are replaced by distinct characters (such as “XXXXXXXX” for example) and by new random values, respectively, in sanitizations. Corresponding hash values are also recomputed. Thus a new hash value set  $H'$  and its signature  $s'$  is obtained. Then the sanitizer publishes the sanitized document and the sanitizer’s signature  $s'$  in addition to the original signature  $s$ . On input these data, a verifier verifies the integrity of the sanitized document. In addition, sanitized blocks can be identified by comparing two sets  $H$  and  $H'$ . Finally, from a disclosed document and the sanitized index sets, the verifier can verify the correctness of the sanitizations. An formal description of the proposed sanitizable signature scheme is in Table 3 (see also Figure 2)D We name the scheme as the Partial Information Assuring Technology for Signature (PIATS).

By the described procedures, a verifier can verify the integrity of disclosed blocks, identify the sanitized parts and sanitizers. If a closed block (combined with former and latter blocks) has a meaning, it can be treated as either a valid sanitization or a forgery depending on the policy. Thus PIATS allows multiple sanitizations with avoiding dishonest sanitizations.

### 3.3 Security Analysis

Let us consider the security of the proposed scheme. The unforgeability (no forgeability of signatures), secrecy (no leakage of sanitized information), and the integrity (no forgery on unsanitized parts with valid verification), is established as in the following observations.

**Unforgeability:** Combined with secure digital signature schemes such as RSA-PSS [PKCS], any forgeries can be avoided in the proposed scheme.

**Secrecy:** Since a signer’s signature  $S$  consists of a set of hash values and its signature, and a sanitizer’s signature  $S'$  consists of an updated set of hash values and its signature, the secrecy of the sanitized information is assured by the preimage resistance of the hash function.

**Integrity:** In the proposed scheme, by comparing a hash value set generated by a signer and by a sanitizer, dishonest sanitizations can be identified by verifiers. This is assured by the 2nd preimage resistance of the hash function.

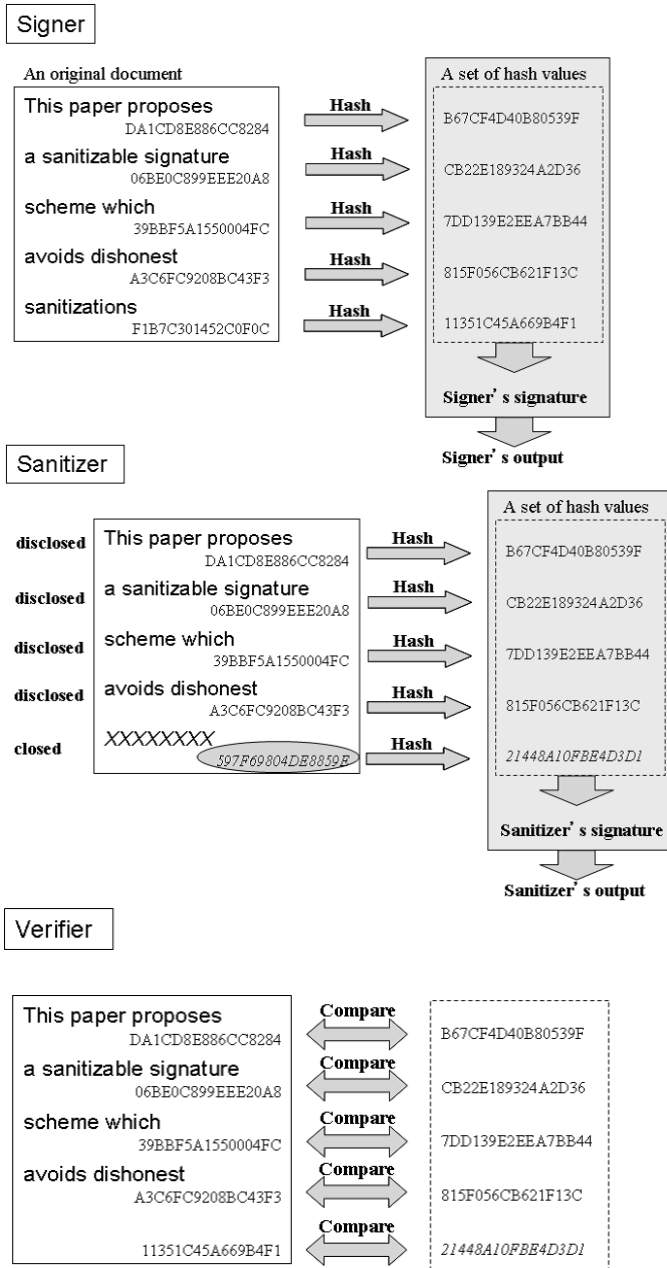


Fig. 2. An outline of the proposed sanitizable signature scheme “PIATS”

**Table 3.** A description of the proposed sanitizable signature scheme “PIATS”**Signer**

- 
1. For a given original document  $\{m_i\}_{1 \leq i \leq n}$ , a signer pads random values  $r_i$  to each block.
  2. For a padded document  $M = \{(m_i, r_i)\}_{1 \leq i \leq n}$ , generate a set of hash values  $H = \{h_i = \text{Hash}(m_i, r_i)\}_{1 \leq i \leq n}$  for a given hash function  $\text{Hash}(\cdot)$ .
  3. Generate a signature  $s = \text{Sign}_{\text{signer}}(H)$  for a signer’s signing function  $\text{Sign}_{\text{signer}}(\cdot)$ .
  4. Set  $S = H || s$  where  $||$  denotes a concatenation.
  5. Output  $(M, S)$  as an original document and a signature.
- 

**Sanitizer**

- 
1. Determine a disclosing index set  $D \subset \{1, \dots, n\}$ , where blocks  $(m_i, r_i)$  ( $i \in D$ ) will be disclosed.
  2. A sanitizer converts the document  $M$  to a new document  $\tilde{M} = \{\tilde{m}_i\}_{1 \leq i \leq n}$ , defined by

$$\tilde{m}_i = \begin{cases} (m_i, r_i) & \text{if } i \in D \\ (m'_i, r'_i) & \text{if } i \notin D, \end{cases}$$

where  $m'_i$  is a distinct characters (such as “XXXXXXXX”) and  $r'_i$  is a random value for padding.

3. Generate a set of hash values  $H' = \{h'_i = \text{Hash}(\tilde{m}_i)\}_{1 \leq i \leq n}$ . Then generate a new signature  $s' = \text{Sign}_{\text{sanitizer}}(H')$  for a signing function  $\text{Sign}_{\text{sanitizer}}(\cdot)$  and set  $S' = H' || s'$ .
  4. Output the sanitized document and a signature  $(\tilde{M}, S')$  with the index set  $D$ .
- 

**Verifier**

- 
1. On input an original signature  $s$ , the sanitized document and a signature  $(\tilde{M}, S')$ , recover  $H$ ,  $s$ ,  $H'$ ,  $s'$ .
  2. Compute  $\text{Verify}_{\text{signer}}(H, s)$  and confirm the integrity of the original document for a verifying function  $\text{Verify}_{\text{signer}}(\cdot)$ .
  3. Compute  $\text{Verify}_{\text{sanitizer}}(H', s')$  and confirm the integrity of the sanitized document and identify the sanitizer for a verifying function  $\text{Verify}_{\text{sanitizer}}(\cdot)$ .
- 

### 3.4 Multiple Sanitization

The proposed scheme allows multiple sanitizations by adding sanitizers’ signatures. Let  $(M^{(0)}, S^{(0)})$  be a pair of the original document and a signer’s signature on  $M^{(0)}$ . Similarly, let  $(M^{(j)}, S^{(j)})$  be a pair of the  $j$ -th sanitized document and the  $j$ -th sanitizer’s signature on  $M^{(j)}$  for  $1 \leq j \leq k$ . Here  $k$  is the admissible number of sanitizers determined in advance as a security parameter of the scheme. Then the last ( $k$ -th) sanitizer publishes  $(M^{(k)}, S^{(0)}, S^{(1)}, \dots, S^{(k)})$ .

From the published information, a verifier can identify which blocks are sanitized by the  $j$ -th sanitizer ( $1 \leq j \leq k$ ) and which blocks are signed by the signer.

The proposed scheme has a property that the number of disclosing blocks can be increasing. This property may be required in most situations where sanitizable signature schemes are used. Conversely and interestingly, the proposed scheme also has a property that the number of disclosing blocks can be decreasing, if all sanitizers can access on the original document.

## 4 Comparison

In this section, we compare sanitizable signature schemes including CES-CV [SBZ01], SUMI-4 [MSI+03], SUMI-5 [MIM+05], and the proposed scheme PIATS from viewpoints of the ability of multiple sanitizations, required conditions for future sanitizations, and combinable signature schemes. A comparison is summarized in Table 4. Note that as described in section 2, CES-CV and CES-HT are potentially identical to SUMI-4.

CES-RSAP and CES-MERP are combined to only specified (RSA-type) signature schemes, while other schemes can be combined with arbitrary secure signature schemes. All sanitizable schemes but SUMI-1 support multiple sanitizations. A main difference between SUMI-2, SUMI-3, SUMI-4 is the number of required signatures; for signing an  $n$ -block document, SUMI-2, SUMI-3, and SUMI-4 requires  $2^n$ ,  $n$ , 1 signatures, respectively. Thus SUMI-4, its successor SUMI-5, and PIATS are efficient with regard to the number of required signatures. Only SUMI-5 and PIATS can avoid dishonest sanitizations. However, as described in section 2, SUMI-5 requires a rather impractical assumption in which a closing policy should be determined in the beginning. On the other hand, PIATS avoids dishonest sanitizations without such a limitation.

**Table 4.** A comparison of sanitizable signature schemes

Scheme	Multiple Sanitization		Combinable Signature Scheme	Specifying Future Sanitizations
	(Honest)	(Dishonest)		
CES-CV CES-HT	Possible	Possible	Arbitrary	Unrequired
CES-RSAP CES-MERP	Possible	Possible	Limited (RSA-type) (RSA-type)	Unrequired
SUMI-1	Impossible	Impossible	Arbitrary	—
SUMI-2	Possible	Possible	Arbitrary	Unrequired
SUMI-3	Possible	Possible	Arbitrary	Unrequired
SUMI-4	Possible	Possible	Arbitrary	Unrequired
SUMI-5	Possible	Partially possible	Arbitrary	Required
<b>PIATS</b>	<b>Possible</b>	<b>Impossible</b>	<b>Arbitrary</b>	<b>Unrequired</b>

## 5 Concluding Remarks

This paper proposes a new partially sanitizable signature scheme PIATS which support multiple sanitizations with avoiding dishonest sanitizations, and therefore enables to manage documents with secure and privacy-protected. Obviously, the proposed scheme is suitable for managing digital documents (described in XML format for example). However, applying PIATS to scanned documents is not easy, because scanned documents are recorded in picture formats. Since there are so many types of formats, we have to consider how to apply PIATS to each format separately. Recently, we developed an experimental system which manages a sanitizable signature schemes on jpeg formats. Intuitively, the system works well, but there are many problems to overcome. Further experiments and analysis will be required to use PIATS in practical systems.

## Acknowledgments

The authors would like to thank Naoya Torii, Satoru Torii and Takeshi Shimoyama for their helpful comments and suggestions on the early version of this paper.

## References

- [ACM+05] G. Ateniese, D.H. Chou, B. Medeiros, G. Tsudik, “Sanitizable Signatures”, *ESORICS 2005*, LNCS 3679, pp. 159–177, Springer-Verlag, 2005.
- [MIM+05] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, H. Imai, “Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control”, *The Institute of Electronics, Information and Communication Engineers (IEICE) Trans. on Fundamentals*, Vol, E88-A, pp. 239–246, No. 1, January 2005. Available from <http://search.ieice.org/index-e.html>
- [MSI+03] K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, “Digital Documents Sanitizing Problem”, *The Institute of Electronics, Information and Communication Engineers (IEICE) technical report*, ISEC 2003-20, May 2003.
- [PKCS] RSA Laboratories, “PKCS #1 v2.1: RSA Encryption standard”, June 14, 2002, Available from <http://www.rsasecurity.com/rsalabs/node.asp?id=2125>
- [SBZ01] R. Steinfeld, L. Bull, and Y. Zheng, “Content Extraction Signatures”, *ICICS 2001*, LNCS 2288, pp. 285–304, Springer-Verlag, 2001.
- [Wir00] “Carniove Review Team Exposed!”, an article of *Wired News Reports*, 2000. Available from <http://www.wired.com/news/politics/0,1283,39102,00.html>
- [Wir02] “NYT Site Exposes CIA Agents”, an article of *Wired News Reports*, 2002. Available from <http://www.wired.com/news/politics/0,1283,37205,00.html>



# Ciphertext Comparison, a New Solution to the Millionaire Problem

Kun Peng<sup>1</sup>, Colin Boyd<sup>1</sup>, Ed Dawson<sup>1</sup>, and Byoungcheon Lee<sup>2</sup>

<sup>1</sup> Information Security Institute,  
Queensland University of Technology,  
{k.peng, c.boyd, e.dawson}@qut.edu.au

<http://www.isi.qut.edu.au>

<sup>2</sup> Joongbu University, Korea

sultan@joongbu.ac.kr

**Abstract.** A new cryptographic protocol —ciphertext comparison— can compare two ciphertexts without revealing the two encrypted messages. Correctness of the comparison can be publicly verified. This technique provides an efficient and publicly verifiable solution to the famous millionaire problem. It is the first solution to the millionaire problem to output a precise result (the two messages are equal or which is larger). Privacy in this new solution is achieved with an overwhelmingly large probability and strong enough in practice.

**Keywords:** Ciphertext comparison, the millionaire problem, efficiency.

## 1 Introduction

In the millionaire problem, two millionaires want to compare their richness without revealing their wealth. This problem can be formulated as a comparison of two ciphertexts without decrypting them. The millionaire problem is an intensively studied problem in multiparty computation. Since this problem was raised by Yao [17], many multiparty computation schemes [13, 12, 6, 11, 3, 16, 2, 8, 15] have been proposed, each of which can be applied to the millionaire problem. However, none of the currently known multiparty computation schemes provides an efficient and verifiable solution to the millionaire problem. Moreover, all the existing solutions to the millionaire problem only output one bit. So they output an imprecise result (whether a message is larger than the other or no larger than the other), while a precise result should indicate a message is larger than the other or equal to the other or smaller than the other. In addition, many of the existing schemes have various other problems like lack of verifiability.

A new protocol proposed in this paper, ciphertext comparison, can efficiently implement comparison of two encrypted messages without revealing them. A distributed homomorphic encryption algorithm is employed to encrypt the two messages. The ciphertext comparison technique outputs  $a(m_1 - m_2)$  where  $a$  is a random and secret integer. Parameter choice guarantees that  $a(m_1 - m_2)$  indicates the comparison result, but does not reveal any information about the two

messages with an overwhelmingly large probability except which one is larger. The whole protocol is publicly verifiable. Correctness, privacy, robustness, public verifiability and high efficiency are achieved simultaneously for the first time in solving the millionaire problem. Moreover, a precise result is output in this new solution.

The rest of this paper is organised as follows. In Section 2, the millionaire problem and its previous solutions are recalled. In Section 3, new primitives needed in this paper are proposed and proved to be secure. In Section 4, the new ciphertext comparison protocol is described. In Section 5, the new ciphertext comparison protocol is analysed and compared against previous solutions to the millionaire problem. The paper is concluded in Section 6.

In the rest of this paper, the following symbols are used.

- $\parallel$  stands for concatenation.
- $\lfloor x \rfloor$  is the largest integer no more than  $x$ .
- $PKN(x_1, x_2, \dots, x_n | cond)$  stands for the proof of knowledge of a set of integers  $x_1, x_2, \dots, x_n$  satisfying a given condition  $cond$ .

## 2 The Millionaire Problem and Related Work

The millionaire problem was raised by Yao [17]. In the millionaire problem, two millionaires want to compare who is richer without revealing their wealth. This problem can be formulated as a comparison of two encrypted messages without revealing them. Some participants (sometimes the two millionaires themselves) are employed to solve the problem without revealing the two messages. The following four properties are often desired in a solution protocol to the millionaire problem.

- Correctness: If the two ciphertexts are decrypted and then compared, the result is the same as the protocol outputs.
- Precision: A precise result must be output to indicate exactly which of the three possibilities (whether a message is smaller than or equal to or larger than the other) occurs.
- Public verifiability: Each participant can be publicly verified to honestly follow the protocol.
- Privacy: After the computation, no information about the two messages is revealed except the comparison result.

Solutions to the millionaire problem always employ multiparty computation. In multiparty computation, multiple participants compute a function with encrypted inputs and determine the result of the function without revealing the inputs. They usually employ an evaluation circuit consisting of some logic gates to compute the function in ciphertext. Usually the decryption key of the employed encryption algorithm is shared among the participants, so that privacy of the encrypted inputs can be protected with an assumption that the number of malicious participants is not over a threshold. In all the known existing multiparty computation solutions, only an imprecise result is output (a precise result should indicate one of the three possible results).

According to the computation in every gate in the circuit, they can be divided into two methods. The first method is based on encrypted truth tables. Namely, the rows in the truth table of each logic gate in the circuit are encrypted and shuffled, so that any legal encrypted input to each gate can be matched to an encrypted output without being revealed. The second method is based on logic homomorphism of certain encryption schemes. As special encryption algorithms homomorphic in regard to the logic gates in the circuit are employed, the evaluation can be implemented by computing in ciphertext without the help of any truth table. The recent schemes employing the first method include [13], [12], [6], [11] and [3]. The recent schemes employing the second method include [16], [2], [8] and [15]. None of them provides a correct, precise, private, verifiable and efficient solution to the millionaire problem. Such a solution will be designed in this paper. The new technique is called ciphertext comparison. It employs the second method, but in a novel manner.

### 3 Preliminary Work

Three cryptographic primitives are presented in this section and will be applied to the new ciphertext comparison protocol. All multiplications in this section are with a modulus  $N^2$  where  $N$  is the Paillier composite [14].

#### 3.1 Proof of Knowledge of $N^{th}$ Root mod $Z_{N^2}^*$

Proof of knowledge of root was proposed by Guillou and Quisquater [10], in which an honest verifier ZK proof of knowledge of  $v^{th}$  root with a composite modulus  $n$  was presented and proved to be secure. A variation of the proof protocol of knowledge of root in [10] is described here. In the protocol in Figure 1, a specific setting is employed: knowledge of  $N^{th}$  root modulo  $N^2$  must be proved where  $N$  is a Paillier composite. The protocol is used to prove the knowledge of  $x$ , the  $N^{th}$  root of  $y$  and an integer in  $Z_{N^2}^*$ , where  $P$  and  $V$  stand for prover and verifier. This proof protocol is consistent with the Paillier setting and can be applied to verify the validity of Paillier encryption. Correctness of this protocol is straightforward. Namely, when the prover knows a  $N^{th}$  root of  $y$ , he can pass the verification. Since the setting is different from the original protocol in [10], it must be proved that the new protocol is sound with Paillier setting.

$P \rightarrow V : b = r^N \text{ where } r \text{ is randomly chosen from } Z_N.$ $V \rightarrow P : e, \text{ where }  e  = 160.$ $P \rightarrow V : w = rx^e$ $\text{Verification: } w^N = by^e$
---

**Fig. 1.** Proof of Knowledge of  $N^{th}$  Root

**Theorem 1.** *The proof protocol of knowledge of  $N^{\text{th}}$  root in Figure 1 is specially sound if  $N$  is correctly generated. More precisely, if the prover can provide correct responses to two different challenges with a same commitment, he can calculate a  $N^{\text{th}}$  root of  $y$  efficiently.*

*Proof:* If the prover can provide responses  $w_1$  and  $w_2$  to a commitment  $b$  and two different challenges  $e_1$  and  $e_2$  where  $e_1 > e_2$ , such that

$$w_1^N = by^{e_1} \quad (1)$$

$$w_2^N = by^{e_2} \quad (2)$$

then (1) divided by (2) yields

$$(w_1/w_2)^N = y^{e_1 - e_2}$$

According to the Euclidean algorithm, integers  $\alpha$  and  $\beta$  can be found, such that  $\beta(e_1 - e_2) = \alpha N + \gcd(N, e_1 - e_2)$ . As  $N = pq$  is correctly generated,  $p$  and  $q$  are primes and the length of  $p$  and  $q$  is much longer than  $|e_1 - e_2|$ , so  $\gcd(N, e_1 - e_2) = 1$ . So

$$(w_1/w_2)^{\beta N} = y^{\beta(e_1 - e_2)} = y^{\alpha N + 1}$$

Namely,

$$y = ((w_1/w_2)^\beta / y^\alpha)^N$$

So,  $(w_1/w_2)^\beta / y^\alpha$  is a  $N^{\text{th}}$  root of  $y$ . Note that the prover can calculate  $\alpha$  and  $\beta$  efficiently from  $N$  and  $e_1 - e_2$  using Euclidean algorithm. Therefore, the prover can get a  $N^{\text{th}}$  root of  $y$  efficiently.  $\square$

**Theorem 2.** *The proof protocol of knowledge of  $N^{\text{th}}$  root in Figure 1 is honest verifier zero knowledge.*

*Proof:* A simulator with no knowledge of any  $N^{\text{th}}$  root of  $y$  can choose  $e$  and  $w$  randomly and calculate  $b = w^N / y^e$ . Thus a simulated transcript composed of uniformly distributed  $b$ ,  $e$  and  $w$  is obtained. The proof transcript generated by a prover with knowledge of an  $N^{\text{th}}$  root of  $y$  and an honest verifier (who chooses the challenge randomly and independently) is also composed of uniformly distributed  $e$ ,  $w$ ,  $b$ . These two transcripts are indistinguishable. So these two proof transcripts are indistinguishable.  $\square$

According to Theorem 1 and Theorem 2, this proof protocol is a so-called  $\Sigma$ -protocol [7]. So according to Damgard's analysis in [7], this proof is sound (the probability that a prover without the knowledge of a  $N^{\text{th}}$  root of  $y$  can pass the verification in this protocol is no more than  $2^{-160}$ ) and private (the prover's knowledge of  $N^{\text{th}}$  root of  $y$  is not revealed). Hash function  $H()$  can be employed to generate the challenge as  $e = H(y||b)$ , so that the protocol becomes non-interactive. In the rest of this paper, non-interactive proof of knowledge of  $N^{\text{th}}$  root is applied. If  $H()$  can be seen as a random oracle, security is not compromised in the non-interactive proof.

### 3.2 Proof of Knowledge of 1-Out-of-2 $N^{th}$ Root mod $Z_{N^2}^*$

The proof protocol in Figure 2 is a combination of the proof of  $N^{th}$  root in Figure 1 and the proof of partial knowledge [5] to prove the knowledge of  $x$ , the  $N^{th}$  root of  $y_1$  or  $y_2$ , integers in  $Z_{N^2}^*$ . For simplicity, it is supposed without losing generality  $x^N = y_2$ . Correctness of this protocol is straightforward. Namely, when the prover knows a  $N^{th}$  root of either  $y_1$  or  $y_2$ , he can pass the verification. As the proof of knowledge of  $N^{th}$  root modulo  $N^2$  in Section 3.1 and the partial proof technique in [5] are both specially sound and honest verifier ZK, this protocol is also specially sound and honest verifier ZK. Namely, the probability that a prover without the knowledge of a  $N^{th}$  root of  $y_1$  or  $y_2$  can pass the verification in this protocol is no more than  $2^{-160}$  the prover's knowledge of  $N^{th}$  root of  $y_1$  or  $y_2$  is not revealed. Moreover, this protocol can also be extended to be non-interactive without compromising its security when a hash function regarded as a random oracle is used to generate the challenge  $e$ .

1. The prover chooses  $r, w_1$  and  $e_1$  randomly from  $Z_N^*, Z_{N^2}^*$  and  $\{0, 1\}^{160}$  respectively. He calculates  $b_1 = w_1^N y_1^{e_1}$  and  $b_2 = r^N$ .
2. The verifier randomly chooses a 160-bit challenge  $e$ .
3. The prover calculates  $e_2 = e - e_1$  and  $w_2 = r/x^{e_2}$ .
4. The prover publishes  $e_1, w_1, e_2$  and  $w_2$ . Anybody can verify  $e = e_1 + e_2$  and  $b_1 = w_1^N y_1^{e_1}$  and  $b_2 = w_2^N y_2^{e_2}$ .

**Fig. 2.** Proof of Knowledge of 1-out-of-2  $N^{th}$  Root

### 3.3 A Combined Proof of Equality of Exponents and Knowledge of $N^{th}$ Root

Let  $g_1, g_2, y_1$  and  $y_2$  be in  $Z_{N^2}^*$ . The proof protocol in Figure 3 is used to prove  $PKN(x, r_1, r_2 \mid x \in Z, r_1 \in Z_N^*, r_2 \in Z_N^*, y_1 = g_1^x r_1^N, y_2 = g_2^x r_2^N)$ . Correctness of this protocol is straightforward. Namely, if the prover knows  $x, r_1, r_2$  and follows the protocol, the verifier will accept his proof. Soundness of this protocol seems at first to be straightforward if it is regarded as a combination of proof of equality of logarithms [4] and proof of knowledge of  $N^{th}$  root in Section 3.1, both

1. The prover chooses  $v \in Z_N, u_1 \in Z_N^*$  and  $n_2 \in Z_N^*$  randomly and calculates  $\gamma = g_1^v u_1^N$  and  $\theta = g_2^v u_2^N$ . He sends  $\gamma$  and  $\theta$  to the verifier.
2. The verifier randomly chooses a 160-bit challenge  $e$  and sends it to the prover.
3. The prover calculates  $z_1 = v - ex, z_2 = u_1/r_1^e, z_3 = u_2/r_2^e$  and sends them to the verifier.
4. The verifier verifies  $\gamma = g_1^{z_1} z_2^N y_1^e$  and  $\theta = g_2^{z_1} z_3^N y_2^e$ . He accepts the proof only if these two equations are correct.

**Fig. 3.** Combined Proof of Equality of Exponent and Knowledge of  $N^{th}$  Root

of which are sound. However, in this protocol,  $g_1$  and  $g_2$  may be in two different cyclic groups with different orders. As the proof of equality of logarithms in [4] can only be applied to prove equality of logarithms in a same group or two groups with a same order, it cannot be applied here. To the authors' knowledge, the only technique to prove equality of logarithms in groups with different orders was proposed by Bao [1]. However, his technique is only sound (passing his verification guarantee two logarithms in different groups with different orders are equal with a very large probability) but not correct (lots of equal logarithm pairs in the two groups cannot pass the verification with a very large probability) so can only be applied to his special application — a verifiable encryption scheme. As our protocol must be both correct and sound, our technique is different from his in that equality of exponents instead of equality of logarithms is proved. Namely, it is not required in our scheme that the two exponents are equal with two different modulus. It is enough that the two exponents are equal without any modulus. Soundness of our protocol is proved in Theorem 3.

**Theorem 3.** *The proof protocol in Figure 3 is specially sound. More precisely, if the prover can provide correct responses for two different challenges to a same commitment, he can efficiently calculate  $x$ ,  $r_1$  and  $r_2$ , such that  $x \in Z$ ,  $r_1 \in Z_N^*$ ,  $r_2 \in Z_N^*$ ,  $y_1 = g_1^x r_1^N$ ,  $y_2 = g_2^x r_2^N$  if  $N$  is correctly generated.*

*Proof:* If the prover can provide two sets of responses  $z_{1,1}$ ,  $z_{2,1}$ ,  $z_{3,1}$  and  $z_{1,2}$ ,  $z_{2,2}$ ,  $z_{3,2}$  for two different challenges  $e_1$  and  $e_2$  and the same commitment pair  $\gamma, \theta$ , such that

$$\gamma = g_1^{z_{1,1}} z_{2,1}^N y_1^{e_1} \quad (3)$$

$$\theta = g_2^{z_{1,1}} z_{3,1}^N y_2^{e_1} \quad (4)$$

$$\gamma = g_1^{z_{1,2}} z_{2,2}^N y_1^{e_2} \quad (5)$$

$$\theta = g_2^{z_{1,2}} z_{3,2}^N y_2^{e_2} \quad (6)$$

(3) divided by (5) yields

$$g_1^{z_{1,1}} z_{2,1}^N y_1^{e_1} = g_1^{z_{1,2}} z_{2,2}^N y_1^{e_2}$$

So,

$$g_1^{z_{1,1}-z_{1,2}} (z_{2,1}/z_{2,2})^N = y_1^{e_2-e_1}$$

(4) divided by (6) yields

$$g_2^{z_{1,1}-z_{1,2}} (z_{3,1}/z_{3,2})^N = y_2^{e_2-e_1}$$

According to the Euclidean algorithm, integers  $\alpha$  and  $\beta$  can be found, such that  $\beta(e_1 - e_2) = \alpha N + \gcd(N, e_1 - e_2)$ . So

$$g_1^{\beta(z_{1,1}-z_{1,2})} (z_{2,1}/z_{2,2})^{\beta N} = y_1^{\alpha N + \gcd(N, e_1 - e_2)}$$

and

$$g_2^{\beta(z_{1,1}-z_{1,2})} (z_{3,1}/z_{3,2})^{\beta N} = y_2^{\alpha N + \gcd(N, e_1 - e_2)}$$

As  $N = pq$  and the  $p$  and  $q$  are primes with length much longer than  $|e_1 - e_2|$  ( $N$  is a correctly generated Paillier composite),  $\gcd(N, e_1 - e_2) = 1$ . So,

$$g_1^{\beta(z_{1,1}-z_{1,2})}((z_{2,1}/z_{2,2})^\beta/y_1^\alpha)^N = y_1 \tag{7}$$

and

$$g_2^{\beta(z_{1,1}-z_{1,2})}((z_{3,1}/z_{3,2})^\beta/y_2^\alpha)^N = y_2 \tag{8}$$

Note that the prover can efficiently calculate  $\alpha$  and  $\beta$  easily from  $N$  and  $e_1 - e_2$  using Euclidean algorithm. Therefore, the prover can get  $x = \beta(z_{1,1} - z_{1,2})$ ,  $r_1 = (z_{2,1}/z_{2,2})^\beta/y_1^\alpha$  and  $r_2 = (z_{3,1}/z_{3,2})^\beta/y_2^\alpha$  efficiently, such that  $x \in Z$ ,  $r_1 \in Z_N^*$ ,  $r_2 \in Z_N^*$ ,  $y_1 = g_1^x r_1^N$ ,  $y_2 = g_2^x r_2^N$ .  $\square$

**Theorem 4.** *The proof protocol in Figure 3 is honest verifier zero knowledge.*

This theorem can be proved like Theorem 2.

According to Theorem 3 and Theorem 4, the proof protocol in Figure 3 is sound (the probability that a prover without the required knowledge can pass the verification in this protocol is no more than  $2^{-160}$ ) and private (the prover’s secret knowledge is not revealed). A hash function  $H()$  can be employed to generate the challenge as  $e = H(y_1||y_2||\gamma||\theta)$ , so that the protocol becomes non-interactive. In the rest of this paper, the non-interactive version of this proof is applied. If  $H()$  can be seen as a random oracle, security is not compromised in the non-interactive proof. Note that this protocol does not guarantee the secret knowledge  $x$  is smaller than  $order(g_1)$  or  $order(g_2)$ . That is why we say that equality of exponents instead of equality of logarithms is included in this protocol.

## 4 Ciphertext Comparison

Suppose two  $L$ -bit messages  $m_1$  and  $m_2$  encrypted in  $c_1$  and  $c_2$  respectively are to be compared. The main idea of the comparison is comparing  $F(m_1)$  and  $F(m_2)$  where  $F()$  is a monotonely increasing one-way function. Based on this idea, a comparison technique  $Com(c_1, c_2)$  can be designed, such that  $Com(c_1, c_2) = 1$  if  $m_1 > m_2$ ;  $Com(c_1, c_2) = 0$  if  $m_1 = m_2$ ;  $Com(c_1, c_2) = -1$  if  $m_1 < m_2$ . The comparison procedure is as follows.

1. An additive homomorphic encryption algorithm with encryption function  $E()$  is employed, such that  $E(x_1 + x_2) = E(x_1)E(x_2)$  and  $E(ax) = E(x)^a$  for any messages  $x$ ,  $x_1$ ,  $x_2$  and factor  $a$ . The public key is published while the private key is shared by participants  $A_1, A_2, \dots, A_m$ . The message space of the encryption algorithm is  $\{0, 1, \dots, N - 1\}$ , where  $2^{L+mL'} < \lfloor N/2 \rfloor$  and  $L'$  is a security parameter.
2.  $m_i$  is encrypted into  $c_i = E(m_i)$  for  $i = 1, 2$ . It is proved that  $c_i$  is an encryption of a message with  $L$  bits without revealing the message for  $i = 1, 2$ .

3. Each  $A_l$  chooses  $a_l$  so that  $a_l \in \{0, 1, \dots, 2^{L'} - 1\}$  and calculates  $c'_l = c'_{l-1}$  for  $i = 1, 2$  where  $c'_0 = c_1/c_2$ .  $A_l$  proves  $\log_{c'_{-1}} c'_l < 2^{L'}$  without revealing  $a_l$  for  $l = 1, 2, \dots, m$ .
4. The authorities cooperate to decrypt  $c'_m$ .

$$Com(c_1, c_2) = \begin{cases} 1 & \text{if } 0 < D(c'_m) \leq \lfloor N/2 \rfloor \\ 0 & \text{if } D(c'_m) = 0 \\ -1 & \text{if } D(c'_m) > \lfloor N/2 \rfloor \end{cases} \quad (9)$$

Any distributed additive homomorphic encryption algorithm can be employed in this ciphertext comparison. In this section, the ciphertext comparison protocol is described in detail based on distributed Paillier (see [9]).

#### 4.1 Bit Encryption and Its Validity Verification

Messages  $m_1$  and  $m_2$  must be encrypted in a special way such that it is publicly verifiable from their encryptions that they are in the range  $\{0, 1, \dots, 2^L - 1\}$ . So the following encryption-by-bit method is employed.

- Paillier encryption with distributed decryption (see [9]) is employed such that the parameters  $N$ ,  $m$ ,  $L'$  and  $L$  satisfy  $2^{L+mL'} < (N-1)/2$  where  $m$  is the number of participants and  $N$  is the Paillier composite.
- Binary representation of  $m_i$  is a vector  $(m_{i,1}, m_{i,2}, \dots, m_{i,L})$  for  $i = 1, 2$  where  $m_{i,j} \in \{0, 1\}$  and  $m_i = \sum_{j=1}^L m_{i,j} 2^{j-1}$ .
- Component  $m_{i,j}$  is encrypted with Paillier encryption to  $c_{i,j} = g^{m_{i,j}} \cdot r_{i,j}^N \bmod N^2$  for  $i = 1, 2$  and  $j = 1, 2, \dots, L$  where  $r_{i,j}$  is randomly chosen from  $Z_N^*$ .
- The encrypted vectors  $(c_{i,1}, c_{i,2}, \dots, c_{i,L})$  for  $i = 1, 2$  are published.
- The encryptor (millionaire or more generally message provider) proves that each  $c_{i,j}$  is an encryption of 0 or 1 by providing a proof of knowledge of  $c_{i,j}^{1/N}$  or  $(c_{i,j}/g)^{1/N}$  for  $i = 1, 2$  and  $j = 1, 2, \dots, L$ . Proof of knowledge of 1-out-of-2  $N^{th}$  root in the Paillier setting described in Section 3.2 is employed in the proof.
- Anybody can verify validity of  $c_{i,j}$  for  $i = 1, 2$  and  $j = 1, 2, \dots, L$ . If the verification is passed, two ciphertexts  $c_i = \prod_{j=1}^L c_{i,j}^{2^{j-1}} \bmod N^2 = g^{\sum_{j=1}^L m_{i,j} 2^{j-1}} \cdot r_i^N \bmod N^2 = g^{m_i} r_i^N \bmod N^2$  for  $i = 1, 2$  are formed for comparison where  $r_i = \prod_{j=1}^L r_{i,j}^{2^{j-1}} \bmod N$ .

Only if the two ciphertexts are verified to be  $L$  bits long, can they be compared.

#### 4.2 The Comparison Function

The authorities  $A_1, A_2, \dots, A_m$  compare  $c_1 = g^{m_1} r_1^N \bmod N^2$  and  $c_2 = g^{m_2} r_2^N \bmod N^2$  and validity of the comparison can be publicly verified.

1.  $a_l$  is selected randomly from  $\{0, 1, \dots, 2^{L'} - 1\}$  while its validity is guaranteed by bit encryption and its validity verification.



- (a)  $A_l$  chooses  $a_l$  randomly from  $\{0, 1, \dots, 2^{L'} - 1\}$  with binary representation  $(a_{l,1}, a_{l,2}, \dots, a_{l,L'})$  where  $a_l = \sum_{j=1}^{L'} a_{l,j} 2^{j-1}$ . He keeps them secret and publishes  $d_{l,j} = g^{a_l} \cdot t_{l,j}^N \bmod N^2$  for  $j = 1, 2, \dots, L'$  where  $t_{l,j}$  is randomly chosen from  $Z_N^*$ .
- (b)  $A_l$  proves that each  $d_{l,j}$  contains 0 or 1 by providing a proof of knowledge of  $N^{th}$  root of either  $d_{l,j}$  or  $d_{l,j}/g$  modulus  $N^2$  as proposed in Section 3.2.
- (c) Anybody can verify the validity of  $d_{l,j}$  for  $j = 1, 2, \dots, L'$  and calculates  $d_l = \prod_{j=1}^{L'} d_{l,j}^{2^{j-1}} \bmod N^2$ , which is a commitment of  $a_l$ .

Only if  $d_l$  for  $l = 1, 2, \dots, m$  are verified to be valid, the comparison continues.

- 2. Each  $A_l$  performs  $c'_l = c'_{l-1} a_l s_l^N \bmod N^2$  for  $l = 1, 2, \dots, m$  where  $c'_0 = c_1/c_2 \bmod N^2$  and  $s_l$  is randomly chosen from  $Z_N^*$ .  $A_l$  has to give a proof

$$PKN(a_l, t_l, s_l \mid a_l \in Z, t_l \in Z_N^*, s_l \in Z_N^*, d_l = g^a t_l^N \bmod N^2, c'_l = c'_{l-1} a_l s_l^N \bmod N^2) \tag{10}$$

where  $t_l = \prod_{j=1}^{L'} t_{l,j}^{2^{j-1}} \bmod N$ . This proof can be implemented using the combined proof of equality of exponent and knowledge of  $N^{th}$  root proposed in Section 3.3 and is called *monotone proof*. Only if the verification is passed, the comparison continues.

- 3. The authorities corporately compute  $Com(c_1, c_2)$  by decrypting  $c'_m$ .
- 4. Result of comparison

$$Com(c_1, c_2) = \begin{cases} 1 & \text{if } 0 < D(c'_m) \leq (N - 1)/2 \\ 0 & \text{if } D(c'_m) = 0 \\ -1 & \text{if } D(c'_m) > (N - 1)/2 \end{cases} \tag{11}$$

## 5 Analysis

The ciphertext comparison technique is analysed and compared against the previous solutions to millionaire problem in this section.

### 5.1 Security and Efficiency Analysis

**Theorem 5.** *The proposed ciphertext comparison protocol is correct and sound. More precisely, assume it is infeasible for any  $A_l$  to find  $s$  and  $t$ , such that  $t \not\equiv 1 \pmod N$  and  $g^s t^N = 1 \pmod{N^2}$ , then iff  $m_1 < m_2 < 2^L$ ,  $Com(E(m_1), E(m_2)) = -1$ ; iff  $m_1 = m_2 < 2^L$ ,  $Com(E(m_1), E(m_2)) = 0$ ; iff  $m_2 < m_1 < 2^L$ ,  $Com(E(m_1), E(m_2)) = 1$ .*

*Proof:* As the combined proof of equality of exponents and knowledge of  $N^{th}$  root in Section 3.3 is sound, *monotone proof* (Formula (10)) guarantees that  $A_l$

knows  $a'_l, t'_l$  and  $s_l$  such that  $a'_l \in Z, t'_l \in Z_N^*, s_l \in Z_N^*, d_l = g^{a'} t'^N \bmod N^2$  and  $c'_l = c'_{l-1} s_l^N \bmod N^2$ . As Paillier encryption algorithm is additive homomorphic,

$$\begin{aligned} D(c'_m) &= D((c_1/c_2)^{\prod_{=1} a'}) = \\ D(c_1^{\prod_{=1} a'} / c_2^{\prod_{=1} a'}) &= D(c_1^{\prod_{=1} a'}) - D(c_2^{\prod_{=1} a'}) \bmod N \\ &= D((g^{m_1} r_1^N)^{\prod_{=1} a'}) - D((g^{m_2} r_2^N)^{\prod_{=1} a'}) \bmod N \\ &= D(g^{m_1} \prod_{=1} a' (r_1^{\prod_{=1} a'})^N) - D(g^{m_2} \prod_{=1} a' (r_2^{\prod_{=1} a'})^N) \bmod N \end{aligned}$$

In Section 4.1, validity of  $d_{l,j}$  for  $j = 1, 2, \dots, L'$  is proved by  $A_l$  using the proof of knowledge of 1-out-of-2  $N^{th}$  root proposed in Section 3.2. As the proof of knowledge of 1-out-of-2  $N^{th}$  root is sound, it is guaranteed that  $A_l$  knows  $a_{l,j}$  and  $t_{l,j}$  for  $j = 1, 2, \dots, L'$ , such that  $t_{l,j} \in Z_N^*, a_{l,j} \in \{0, 1\}$  and  $d_{l,j} = g^a \cdot t_{l,j}^N \bmod N^2$ . As  $d_l = \prod_{j=1}^{L'} d_{l,j}^{-1} \bmod N^2$ ,  $A_l$  knows  $a_l = \sum_{j=1}^{L'} a_{l,j} 2^{j-1}$  and  $t_l = \prod_{j=1}^{L'} t_{l,j}^{2^{j-1}} \bmod N$ , such that  $t_l \in Z_N^*, a_l < 2^{L'}$  and  $d_l = g^a t_l^N \bmod N^2$ . Therefore,  $g^{a'} t'^N = g^a t_l^N \bmod N^2$ . Namely,  $g^{a'-a} (t'_l/t_l)^N = 1 \bmod N^2$ . As a result,  $t'_l = t_l \bmod N$ , otherwise  $A_l$  can find  $t'_l/t_l$  and  $a'_l - a_l$ , such that  $t'_l/t_l \neq 1 \bmod N$  and  $g^{a'-a} (t'_l/t_l)^N = 1 \bmod N^2$ , which is contradictory to the assumption that it is infeasible to find  $s$  and  $t$ , such that  $t \neq 1 \bmod N$  and  $g^s t^N = 1$  without knowledge of factorization of  $N$ . So  $a_l = a'_l \bmod \text{order}(g)$ . Therefore,

$$\begin{aligned} D(c'_m) &= D(g^{m_1} \prod_{=1} a' (r_1^{\prod_{=1} a'})^N) - D(g^{m_2} \prod_{=1} a' (r_2^{\prod_{=1} a'})^N) \bmod N \\ &= D(E(m_1 \prod_{l=1}^m a_l)) - D(E(m_2 \prod_{l=1}^m a_l)) \bmod N \\ &= m_1 \prod_{l=1}^m a_l - m_2 \prod_{l=1}^m a_l \bmod N \end{aligned}$$

Since it has been publicly verified that  $2^{L+mL'} < (N-1)/2$ ,  $m_i \in \{0, 1, \dots, 2^L - 1\}$  and  $a_l \in \{0, 1, \dots, 2^L - 1\}$  for  $l = 1, 2, \dots, m$ , function  $F(m_i) = m_i \prod_{l=1}^m a_l$  is monotonely increasing and smaller than  $\lfloor (N-1)/2 \rfloor$ . Therefore, if  $m_1, m_2 < 2^L$ ,

$$D(c'_m) \begin{cases} \in (0, (N-1)/2] & \text{iff } m_1 > m_2 \\ = 0 & \text{iff } m_1 = m_2 \\ > (N-1)/2 & \text{iff } m_1 < m_2 \end{cases} \quad (12)$$

□

The assumption that it is infeasible for  $A_l$  to find  $s$  and  $r$ , such that  $r \neq 1 \bmod N$  and  $g^s r^N = 1 \bmod N^2$  without knowledge of factorization of  $N$  is correct because it seems reasonable to assume that given a constant  $z$  it is infeasible to find  $x$  and  $y$ , such that  $f_1(x)f_2(y) = z$  where  $f_1()$  and  $f_2()$  are one-way functions. As factorization of  $N$  is kept secret to any single authority, both  $f_1(x) = g^x \bmod N^2$  and  $f_2(y) = y^N \bmod N^2$  are one-way functions to  $A_l$ . Moreover, if this assumption is incorrect, any Paillier ciphertext can be decrypted into multiple different messages, which is contradictory to the wide belief that Paillier encryption is secure. Therefore, this assumption is reliable.

The encryption verification in Section 4.1 is private as illustrated in Section 3.2. Privacy of the comparison in Section 4.2 is analysed as follows. If  $A_l$  does not reveal  $a_l$ , it is computationally infeasible for any other party to get any information about  $a_l$  from  $d_l$  as Paillier encryption is secure when the number of dishonest authorities is not over the threshold. Validity proof of  $d_l$  is private as it is a proof of knowledge of 1-out-of-2  $N^{th}$  root proposed in Section 3.2. *Monotone proof* (Formula (10)) is private as it is a combined proof of knowledge of 1-out-of-2  $N^{th}$  root and equality of exponents proposed in Section 3.3. So  $a_l$  is not revealed in these two proofs. So  $m_1 - m_2$  is not revealed from  $D(c'_m)$ , which is equal to  $\prod_{l=1}^m a_l(m_1 - m_2) \bmod N$ . Therefore, none of  $m_1$ ,  $m_2$  or  $m_1 - m_2$  are revealed. However, when  $D(c'_m)$  is too near to the boundaries of its value domain (in  $(0, 2^L)$ ,  $(N - 2^L, N)$ ,  $(2^{mL'}, 2^{L+mL'})$  or  $(N - 2^{L+mL'}, N - 2^{mL'})$ ) partial information is revealed from  $m_1 - m_2$ . The revelation of partial information is demonstrated in Table 1. An example is given in Table 1, where  $N$  is 1024 bits long (according to the widely accepted security standard) and  $L = 40$  (large enough for practical applications). As illustrated in Table 1,  $D(c'_m)$  is usually far away from the boundaries and is in the four special ranges with an overwhelmingly small probability. So, the ciphertext comparison protocol is private with an overwhelmingly large probability. Therefore, the whole ciphertext comparison protocol is private with an overwhelmingly large probability.

**Table 1.** Partial information revelation from  $m_1 - m_2$

	Phenomenon	Revelation	Probability	
			value	example
Case 1	$(c'_m) \in (0, 2^L)$	$m_1 - m_2 \in (0, (c'_m))$	$2^{-L}$	$2^{-984}$
Case 2	$(c'_m) \in (N - 2^L, N)$	$m_2 - m_1 \in (0, (c'_m))$	$2^{-L}$	$2^{-984}$
Case 3	$(c'_m) \in (2^{mL'}, 2^{L+mL'})$	$m_1 - m_2 \in [(c'_m)/2^{mL'}, 2^L)$	$2^{-L}$	$2^{-984}$
Case 4	$(c'_m) \in (N - 2^{L+mL'}, N - 2^{mL'})$	$m_2 - m_1 \in [(c'_m)/2^{mL'}, 2^L)$	$2^{-L}$	$2^{-984}$
Totally			$2^{-4L}$	$2^{-982}$

As the encryption verification in Section 4.1 is a proof of knowledge of 1-out-of-2  $N^{th}$  root proposed in Section 3.2, it is sound, namely a ciphertext containing an invalid message can pass the verification with a negligible probability. So the ciphertext comparison protocol is robust. As each step in the ciphertext comparison protocol is publicly verifiable, public verifiability is achieved. As no complex circuit is used, the ciphertext comparison scheme is quite efficient.

### 5.2 Comparison

A comparison between the new solution to the millionaire problem and the existing solutions is provided in Table 2, where the modular multiplications are counted in regard to computation and transportation of integers with significant length (e.g. 1024 bits long) is counted in regard to communication. The

schemes in [6] and [2] are similar to [11] and [16] respectively, so are not analysed separately.  $K$  is the full-length of exponent,  $t$  is the cutting factor in the cut-and-choose mechanism in [13] and [12],  $\lambda$  is a parameter in [8] and  $T$  is a parameter in [15]. An example is used in Table 2, where fair values are chosen for the parameters:  $|N| = 1024$ ,  $K = 1024$ ,  $m = 3$ ,  $L = 100$ ,  $L' = 10$ ,  $t = 40$ ,  $\lambda = 40$  and  $T = 20$ . According to this comparison, the proposed ciphertext comparison technique is the only efficient, publicly verifiable, private and precise solution to millionaire problem.

**Table 2.** Property comparison

	Public	Precise	Computation		Communication	
	verifiability	result	cost	example	cost	example
[13]	No	No	$\geq 15$	$\geq 40960000$	$\geq 37 + 2$	$\geq 148080$
[12]	Yes	No	$\geq 15$	$\geq 40960000$	$\geq 37 + 2$	$\geq 148080$
[11]	Yes	No	average 4665 + 6	477696006	average 1626 + 6	162606
[3]	Yes	No	average $\geq 4039.5$	$\geq 413644800$	$\geq 1543$	$\geq 154300$
[16]	No	No	4	100000000	$\geq 343^3$	$\geq 343000000$
[8]	No	No	$(1.5 ( + 3)) + ( + 1) ( + 1))/2$	516050	$( + 2)$	4200
[15]	Yes	No	$( + 2)( - 1)(1 + 0.5 ) + 37.5$	4052058	25	2500
Proposed	Yes	Yes	$1.5 (5 - ' + 8 + 10 )$	1803264	$5 - ' + 4 + 10$	1162

## 6 Conclusion

A new cryptographic technique —ciphertext comparison— is proposed to compare two ciphertexts and determine which contains a larger message. This new technique is the only efficient and publicly verifiable solution to the millionaire problem. It is also the only precise solution to the millionaire problem. In the new scheme privacy of the two messages is protected with an overwhelmingly large probability.

## References

1. Feng Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *the Smart Card Research Conference, CARDIS'98*, volume 1820 of *Lecture Notes in Computer Science*, pages 213–220, Berlin, 1998. Springer-Verlag.
2. D. Beaver. Minimal-latency secure function evaluation. In *EUROCRYPT '00, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 335–350, Berlin, 2000. Springer.
3. Christian Cachin and Jan Camenisch. Optimistic fair secure computation (extended abstract). In *CRYPTO '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 94–112, Berlin, 2000. Springer-Verlag.
4. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Berlin, 1992. Springer-Verlag.

5. R. Cramer, I. B. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.
6. Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multipart computation from threshold homomorphic encryption. In *EUROCRYPT '01, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299, Berlin, 2001. Springer.
7. Ivan Damgård and Ronald Cramer. On  $\Sigma$ -protocols. *Cryptologic Protocol Theory*, 2002. Available as <http://www.daimi.au.dk/~ivan/Sigma.ps>.
8. Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 457–472, Berlin, 2001. Springer.
9. Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography 2000*, pages 90–104, Berlin, 2000. Springer-Verlag. *Lecture Notes in Computer Science* 1962.
10. L. C. Guillou and J. J. Quisquater. A “paradoxical” identity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Berlin, 1989. Springer-Verlag.
11. M Jakobsson and A Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 143–161, Berlin, 2000. Springer-Verlag.
12. A. Juels and M. Szydlo. A two-server, sealed-bid auction protocol. In *The Sixth International Conference on Financial Cryptography 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 72–86, Berlin, 2002. Springer-Verlag.
13. Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy perserving auctions and mechanism design. In *ACM Conference on Electronic Commerce 1999*, pages 129–139, 1999.
14. P Paillier. Public key cryptosystem based on composite degree residuosity classes. In *EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Berlin, 1999. Springer-Verlag.
15. Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. An efficient and verifiable solution to the millionaire problem. In *Pre-Proceedings of ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 315–330, Berlin, 2004. Springer-Verlag.
16. Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for  $NC^1$ . In *40th Annual Symposium on Foundations of Computer Science, New York, NY, USA, FOCS '99*, pages 554–567, 1999.
17. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *IEEE Symposium on Foundations of Computer Science 1982, FOCS 1982*, pages 160–164, 1992.

# Private Itemset Support Counting

Sven Laur<sup>1</sup>, Helger Lipmaa<sup>2,3</sup>, and Taneli Mielikäinen<sup>4</sup>

<sup>1</sup> Helsinki University of Technology, Finland

<sup>2</sup> Cybernetica AS, Estonia

<sup>3</sup> University of Tartu, Estonia

<sup>4</sup> University of Helsinki, Finland

**Abstract.** Private itemset support counting (PISC) is a basic building block of various privacy-preserving data mining algorithms. Briefly, in PISC, Client wants to know the support of her itemset in Server's database with the usual privacy guarantees. First, we show that if the number of attributes is small, then a communication-efficient PISC protocol can be constructed from a communication-efficient oblivious transfer protocol. The converse is also true: any communication-efficient PISC protocol gives rise to a communication-efficient oblivious transfer protocol. Second, for the general case, we propose a computationally efficient PISC protocol with linear communication in the size of the database. Third, we show how to further reduce the communication by using various tradeoffs and random sampling techniques.

**Keywords:** privacy-preserving data mining, private frequent itemset mining, private itemset support counting, private subset inclusion test.

## 1 Introduction

Frequent itemset mining—also known as frequent pattern mining—is a central task in data mining that has driven research in data mining for ten years. Nowadays, there are special workshops on various aspects of frequent itemset mining [BGZ04]. The goal in frequent itemset mining is to find all frequent itemsets in a given transaction database. Many kinds of data can be viewed as transaction databases and various data mining tasks arising in document analysis, web mining, computational biology, software engineering and so on can be modelled as frequent itemset mining. For example, one can use frequent itemset mining to find which items are usually bought together in a supermarket, or to analyse the correlation between various patterns in the genome database. The mining of frequent itemsets is a very challenging problem, and it is clearly even more challenging in the scenarios when one encounters privacy issues. Several researchers have studied the distributed case with multiple servers, all having a part of the database, who need to mine frequent itemsets in the joint database without revealing each other too much extra information.

We are concerned with a slightly different scenario where the database is owned by a single party, Server, who sells the result of frequent itemset mining (either the collection of all frequent itemsets or the support of a fixed itemset) to others. That is, we consider the itemset support counting (ISC) problem, which is often used as a building block of

frequent itemset mining or association rules mining, but is important also by itself. As an example of ISC, Server could maintain a commercial citation database, and Client could want to find out how many people cite both herself and Shannon. Other possible examples include Internet search engines, mining in medical databases, etc. In most of such applications, some form of privacy must be guaranteed. On the one hand, it is not in Server's interests that Client obtains more information than she has paid for; moreover, in some cases like the medical databases, giving out more information might even be illegal. On the other hand, Client also does not necessarily want Server to know which itemset interests her.

To define the private itemset support counting, let us first describe the setting more formally. Server owns a  $m \times n$  Boolean database  $\mathcal{D}$  that can be considered as a multiset of  $m$  subsets of the set  $[n] = \{1, \dots, n\}$ . Every row in  $\mathcal{D}$  is called a transaction; it might correspond to a transaction in supermarket, with  $j \in \mathcal{D}[i]$  if  $j$ th item was purchased during the  $i$ th transaction. A subset of  $[n]$  is called an itemset, it corresponds to the set of items that can be in the  $i$ th transaction (e.g., the set of items that were bought together). The goal of Client is to determine the support  $\text{supp}_{\mathcal{D}}(\mathcal{Q}) := |\{i : \mathcal{Q} \subseteq \mathcal{D}[i]\}|$  of an itemset  $\mathcal{Q} \subseteq [n]$  in  $\mathcal{D}$ , i.e., to find out how many of Server's transactions contain  $\mathcal{Q}$ . In an  $(m \times n)$ -private itemset support counting (PISC) protocol, Client retrieves  $\text{supp}_{\mathcal{D}}(\mathcal{Q})$ , so that (1) she will get no other information about the database  $\mathcal{D}$  (server-privacy) and (2) Server gets no information about  $\mathcal{Q}$  (client-privacy). In the scope of this paper, we require server-privacy to be information-theoretical and client-privacy to be computational.

The data mining setting implies a few non-standard considerations, mostly due to the large amounts of the handled data. First,  $m$  and  $n$  can be very large (e.g.,  $m, n \geq 10\,000$ ), so whenever possible, it is desirable to have communication and computation of order  $o(mn)$ . Second, again due to the large amount of data, it is impractical to have protocols that are verifiable or even provide correctness. Therefore, we only focus on the privacy issues of the PISC protocols. Thus, we use relaxed security definitions, standard in the case of computationally-private information retrieval, oblivious transfer and oblivious keyword search protocols, where the security of the client is only defined by requiring that his query will remain private. Moreover, we construct protocols that are private in the semi-honest model since they are usually efficient and may suffice in the practice. Protocols, private in the malicious model, can be constructed by adding standard zero-knowledge proofs. In all cases, we put emphasis both on the efficiency of the protocols and on the provable security.

First, we show a close correspondence between PISC and CPIR by providing tight two-way reductions between these two problems. More precisely: (a) Given a  $\binom{s}{1}$ -CPIR protocol CPIR of  $\ell$ -bit strings with communication  $C_{\text{CPIR}}(s, \ell)$ , we show how to construct a  $(2^n \times n)$ -PISC protocol CPIR-PISC with communication  $C_{\text{CPIR}}(2^n, n)$ . Taking the recent  $\binom{s}{1}$ -oblivious transfer protocol for  $\ell$ -bit strings of Lipmaa [Lip05] with communication  $\Theta(\log^2 s + \ell \cdot \log s)$ , this results in communication  $\Theta(n^2)$ . (The use of a very recent CPIR protocol by Gentry and Ramzan [GR05] results in communication  $\Theta(n)$ .) However, in the case of CPIR-PISC, Server needs to store a table of  $2^n \cdot n$  bits and then execute the CPIR protocol on  $2^n$  elements, which is infeasible when say  $n \geq 20$ , while in a realistic data mining application,  $n$  might be larger than 10 000.

(b) Given a  $(m \times n)$ -PISC protocol PISC with communication  $C_{\text{PISC}}(m, n)$ , we show how to construct a  $\binom{2^n - 1}{1}$ -CPIR protocol PISC-CPIR on 1-bit strings with communication  $C_{\text{PISC}}(2^n, n)$ . This enables us to carry over several standard results on the CPIR and oblivious transfer protocols to the PISC scenario. Moreover, the reductions increase communication at most by factor of  $n$ , therefore the optimal communication of the CPIR and PISC protocols can differ only by a logarithmic term in the database size.

For databases with many attributes, we describe an alternative  $(m \times n)$ -PISC protocol PSI-PISC that uses a new *private subset inclusion protocol* PSI, also described in this paper, as a subroutine. The resulting protocol has communication  $(n + m + 1) \cdot k$ , where  $k$  is the bit-length of a ciphertext, and is private in the semi-honest model assuming that the used homomorphic cryptosystem (a) has plaintext space with prime cardinality, and (b) is IND-CPA secure; the Decisional Diffie-Hellman Assumption is sufficient here. The protocol can be made secure in the malicious model by using standard (non-interactive) zero-knowledge proofs. The PSI-PISC protocol is computationally feasible even when  $n \approx 10\,000$ , since the computational work of Server is of order  $\Theta(n + m + w(\mathcal{D}))$  encryptions and decryptions, where  $w(\mathcal{D})$  is the number of 1-bits in the usually very sparse database  $\mathcal{D}$ .

In addition, we study imprecise protocols: we discuss the problem of just detecting whether the given itemset is frequent and study sampling techniques. Random sampling of the database and approximating the itemset support based on the support in the sample allows us to cheaply extend the PSI-PISC protocol to huge databases, supposing that Client is willing to accept approximate answers.

## 2 Preliminaries

For an integer  $s$ , denote  $[s] := \{1, 2, \dots, s\}$ . For a nonnegative integer  $X$ , let  $\text{len}(X) := \lceil \log_2(X + 1) \rceil$  denote the number of bits it takes to store and transfer  $X$ . The statistical difference of two distributions  $X$  and  $Y$  over the discrete support  $Z$  is defined as  $\text{Dist}(X||Y) := \max_{S \subseteq Z} |\Pr[X \in S] - \Pr[Y \in S]|$ .

**Data mining setting.** Our setting is the following, very common one in data mining. The Server has a transaction database  $\mathcal{D}$  over  $n$  attributes (or items)  $A_1, A_2, \dots, A_n$  and the database consists of  $m$  transactions. A transaction is a subset of the attributes. Alternatively, a transaction database  $\mathcal{D}$  of  $m$  transactions over  $n$  attributes can be considered as a  $m \times n$  binary matrix  $\mathcal{D}$  where the entry  $(i, j)$  is one iff  $A_j \in \mathcal{D}[i]$ . In a realistic setting, the resulting 0-1 matrix can have e.g. 100 000 transactions (rows) and 50 000 attributes (columns).

The frequent itemset mining task is, given a transaction database  $\mathcal{D}$  of  $m$  rows and a minimum frequency threshold  $\sigma \in (0, 1]$ , to find the subsets of attributes that are contained in  $\sigma$ -fraction of the transactions, i.e., to determine the collection  $\mathcal{F} = \{X \subseteq \{A_1, \dots, A_n\} : \text{freq}_{\mathcal{D}}(X) \geq \sigma\}$  of  $\sigma$ -frequent itemsets in  $\mathcal{D}$  where  $\text{freq}_{\mathcal{D}}(X) = |\{i \in [m] : X \subseteq \mathcal{D}[i]\}| / m = \text{supp}_{\mathcal{D}}(X) / m$ . Alternatively, the set of frequent itemsets can be specified by the support threshold as  $\mathcal{F} = \{X \subseteq \{A_1, \dots, A_n\} : \text{supp}_{\mathcal{D}}(X) \geq \sigma \cdot m\}$ . Usually also the frequencies or the sup-



ports of the frequent itemsets are required. We assume that attribute labels  $A_1, \dots, A_n$  are public and thus can be substituted with canonical labelling  $\{1, \dots, n\}$ .

Although frequent itemset mining can be done in time  $\mathcal{O}(mn\kappa)$  [AMS<sup>+</sup>96], where  $\kappa$  is the number of frequent itemsets, the running time can easily become intractable, since  $\kappa$  itself is exponential in the cardinality of the largest frequent itemset due to the monotonicity of the support. Therefore, various output compaction techniques are known from the literature, see [BGZ04] for an up-to-date overview of frequent itemset mining algorithms and references.

**Sampling bounds.** Let  $X_1, \dots, X_k$  be independent random 0-1 variables that are drawn from the same distribution with the expectation  $\mu = \Pr[X_i = 1]$ . Let  $X$  be the average  $(X_1 + \dots + X_k)/k$ . Then the Chernoff bound  $\Pr[(1 - \varepsilon)\mu \leq X \leq (1 + \varepsilon)\mu] \leq 2 \cdot \exp\left(-\frac{k\mu\varepsilon^2}{4}\right)$  describes the distribution of relative error and the Hoeffding bound  $\Pr[|X - \mu| \geq \varepsilon] \leq 2 \cdot \exp(-2k\varepsilon^2)$  the distribution of absolute error.

**IND-CPA secure homomorphic cryptosystems.** A public-key cryptosystem is a triple  $\Pi = (G, E, D)$ , where  $G$  is the key generation algorithm that returns  $(\text{sk}, \text{pk})$  consisting of a secret key  $\text{sk}$  and a public key  $\text{pk}$ ,  $E$  is the encryption algorithm and  $D$  is the decryption algorithm. For a fixed public-key cryptosystem  $\Pi$  and a secret key  $\text{sk}$ , let  $\mathcal{C}$  be the ciphertext space, let  $\mathcal{R}$  be the randomness space and let  $\mathcal{P}$  be the plaintext space. Then,  $E_{\text{pk}} : \mathcal{P} \times \mathcal{R} \rightarrow \mathcal{C}$  and  $D_{\text{sk}} : \mathcal{C} \rightarrow \mathcal{P}$ . Define  $\text{Adv}_{\Pi}^{\text{indcpa}}(A) := 2 \cdot |\Pr[(\text{sk}, \text{pk}) \leftarrow G, (m_0, m_1) \leftarrow A(\text{pk}), b \leftarrow \{0, 1\} : A(m_0, m_1, E_{\text{pk}}(m_b; \mathcal{R})) = b] - \frac{1}{2}|$ . We say that  $\Pi$  is  $(\tau, \varepsilon)$ -secure in the sense of IND-CPA if  $\text{Adv}_{\Pi}^{\text{indcpa}}(A) \leq \varepsilon$  for any probabilistic algorithm  $A$  that works in time  $\tau$ .

Cryptosystem  $\Pi$  is *homomorphic* if for any key pair  $(\text{sk}, \text{pk})$ , any  $m, m' \in \mathcal{P}$  and any  $r, r' \in \mathcal{R}$ ,  $E_{\text{pk}}(m; r) \cdot E_{\text{pk}}(m'; r') = E_{\text{pk}}(m + m'; r \circ r')$ , where  $+$  is a group operation in  $\mathcal{P}$ , and  $\circ$  is a groupoid operation in  $\mathcal{R}$ . A few homomorphic cryptosystems are proven to be secure in the sense of IND-CPA under reasonable complexity assumptions.

**Definitions of Client and Server privacy.** Assume that Client and Server want to securely compute a functionality  $f$ , so that Client receives  $f(\mathcal{Q}, \mathcal{D})$  and Server receives nothing, where  $\mathcal{Q}$  is Client's private input and  $\mathcal{D}$  is Server's private input. In our case, Server's input is potentially so huge that all currently known cryptographic techniques fail to provide correctness in tractable time. Therefore, we consider only privacy issues, i.e., we use relaxed security definitions. Thus, we do not require Server to commit to or even "know" a database to which Client's search is effectively applied. Such a relaxation is standard in the case of protocols like oblivious transfer, computationally-private information retrieval and oblivious keyword search; our security definitions correspond closely to the formalisation given in [FIPR05]. Moreover, in a semi-honest case, all proposed protocols have two messages and therefore, standard security definitions can be somewhat simplified.

Denote by Client an honest Client and by Server an honest Server. Let  $\text{Client}_{\text{sk}}(\cdot; \cdot)$  denote Client's (first) message and  $\text{Server}_{\text{pk}}(\cdot; \cdot)$  denote Server's (second) message. Let  $\mathcal{R}_{\text{Client}}$  (resp.  $\mathcal{R}_{\text{Server}}$ ) be the randomness space of an honest Client (resp. Server). Then we say that a two-message protocol  $\Pi$  is  $(\tau, \varepsilon)$ -client-private (in the malicious model), if for any probabilistic algorithm  $A$  with the working time  $\tau$ ,  $\text{Adv}_{\Pi}^{\text{c-privacy}}(A) :=$

$2 \cdot \max_{(\mathcal{Q}_0, \mathcal{Q}_1)} |\Pr [b \leftarrow \{0, 1\} : A(\mathcal{Q}_0, \mathcal{Q}_1, \text{Client}(\mathcal{Q}_b; \mathcal{R}_{\text{Client}})) = b] - \frac{1}{2}| \leq \varepsilon$ . Here,  $\mathcal{Q}_0$  and  $\mathcal{Q}_1$  are assumed to be valid client-side inputs, and the probability is taken over the coin tosses of Client,  $A$  and over the choices of  $b$ .

We define information-theoretical server-privacy in the semihonest model by requiring that for every unbounded honest-but-curious algorithm  $A$ , one can define a simulator Sim that, given solely  $A$ 's private input  $\mathcal{Q}$ ,  $A$ 's random coins  $r$ , and  $A$ 's private output  $f(\mathcal{Q}, \mathcal{D})$ , generates output that is statistically indistinguishable from the view  $(\text{msg}_1, \text{msg}_2)$  of  $A$  that reacts with the honest Server, where  $\text{msg}_1 \leftarrow A(\mathcal{Q}; r)$  and  $\text{msg}_2 \leftarrow \text{Server}(\mathcal{D}, \text{msg}_1; \mathcal{R}_{\text{Server}})$ . More precisely, the advantage of  $A$  is defined  $\text{Adv}_{\Pi}^{\text{server-privacy}}(A) := \max_{(\mathcal{Q}, \mathcal{D})} \text{Dist}(\text{Sim}_{\text{pk}}(\mathcal{Q}, r, f(\mathcal{Q}, \mathcal{D})) || (\text{msg}_1, \text{msg}_2))$ . Here,  $\mathcal{D}$  is assumed to be a valid Server-side input. Protocol is  $\varepsilon$ -server-private (in the semihonest model), if for all unbounded honest-but-curious  $A$ ,  $\text{Adv}_{\Pi}^{\text{server-privacy}}(A) < \varepsilon$ . Security in the malicious model is defined as usually.

**Computationally-private information retrieval (CPIR) and oblivious transfer (OT).** During a single-server  $\binom{m}{1}$ -computationally-private information retrieval protocol, Client fetches  $\mathcal{D}[\mathcal{Q}]$  from the database  $\mathcal{D} = (\mathcal{D}[1], \dots, \mathcal{D}[m])$ ,  $\mathcal{D}[i] \in \mathbb{Z}_{\ell}$  for some fixed domain  $\mathbb{Z}_{\ell}$ , so that a computationally bounded Server does not know which entry Client is learning. In the case of a two-message CPIR protocol, we can use the previously given client-privacy definition. An  $(\tau, \varepsilon)$ -client-private  $\binom{m}{1}$ -CPIR is an (computationally)  $(\tau, \varepsilon)$ -client-private and (information-theoretically)  $\varepsilon'$ -server-private  $\binom{m}{1}$ -OT protocol if it is additionally  $\varepsilon'$ -server-private. A recent  $\binom{m}{1}$ -CPIR protocol by Lipmaa [Lip05],  $\binom{m}{1}$ -LipmaaCPIR, has asymptotic communication  $\Theta(\log^2 m + \log m \cdot \ell)$  (assuming that the security parameter is a constant). Based on the Aiello-Ishai-Reingold CPIR-to-OT transform [AIR01], Lipmaa also described an  $\binom{m}{1}$ -OT protocol with the same asymptotic communication. Lipmaa's protocols are client-private assuming that the underlying Damgård-Jurik homomorphic cryptosystem is IND-CPA secure, or equivalently, if the Decisional Composite Residuosity Problem is hard. Lipmaa's protocols are unconditionally server-private. A very recent  $\binom{m}{1}$ -CPIR protocol by Gentry and Ramzan [GR05] has communication  $\Theta(\log m + \ell)$ .

**Private Keyword Search.** In many data-mining applications, the data is indexed by a relatively small subset of keys  $\mathcal{K} \subseteq \{1, \dots, m\}$ , where the set  $\mathcal{K}$  itself is private. Therefore, if a Client wants to privately access  $\mathcal{D}[\mathcal{Q}]$  the straightforward solution, a  $\binom{m}{1}$ -OT to the database where empty slots are filled with dummy elements is suboptimal. Several solutions that improve communication and computation costs in this situation [CGN97, OK04, FIPR05] have been proposed. Such solutions either combine hash tables and oblivious transfer, or use oblivious evaluation of pseudo-random functions.

### 3 Basic Cryptographic Tool: Private Subset Inclusion Protocol

In a private subset inclusion (PSI) protocol, Client has a set  $\mathcal{Q} \subseteq [n]$ , Server has a set  $\mathcal{S} \subseteq [n]$ , and Client must establish whether  $\mathcal{Q} \subseteq \mathcal{S}$  or not without neither of the parties obtaining any additional information. More precisely, the protocols must satisfy client-privacy and server-privacy as formalised in Sect. 2, where for the ease of implementation we define  $f(\mathcal{Q}, \mathcal{S}) = 0$ , if  $\mathcal{Q} \subseteq \mathcal{S}$ , and  $f(\mathcal{Q}, \mathcal{S}) \neq 0$ , otherwise. We use the

PRIVATE INPUT: Client has a set  $\mathcal{Q}$  and Server has a set  $\mathcal{S}$ .

PRIVATE OUTPUT: Client knows whether  $\mathcal{Q} \subseteq \mathcal{S}$  or not.

**Message 1, Client** Generate a new key pair  $(\text{sk}, \text{pk}) \leftarrow G$ . Send  $\text{pk}$  to Server.

For any  $i \in [n]$ , generate a new nonce  $r_i \leftarrow_r \mathcal{R}$ . Send  $c_i \leftarrow E_{\text{pk}}(\mathcal{Q}[i]; r_i)$  to Server.

**Message 2, Server** Draw  $s \leftarrow_r \mathcal{P}$ ,  $r \leftarrow_r \mathcal{R}$  uniformly at random. Set  $c \leftarrow (\prod_{i:\mathcal{S}[i]=0} c_i)^s \cdot E_{\text{pk}}(0; r)$ . Send  $c$  to Client.

**Post-processing by Client** Set  $t \leftarrow D_{\text{sk}}(c)$ . Accept that  $\mathcal{Q} \subseteq \mathcal{S}$  iff  $t = 0$ .

### Protocol 1. Private homomorphic subset inclusion test protocol

fact that  $\mathcal{Q} \subseteq \mathcal{S} \iff |\mathcal{Q} \cap \mathcal{S}| = |\mathcal{S}|$ . Let  $\mathcal{Q}$  (resp.  $\mathcal{S}$ ) also denote the characteristic function of  $\mathcal{Q}$  (resp.  $\mathcal{S}$ ). That is,  $\mathcal{Q}[i] = 1 \iff i \in \mathcal{Q}$  and  $\mathcal{S}[i] = 1 \iff i \in \mathcal{S}$ .

To solve PSI, we could use a recent private set intersection cardinality protocol by Freedman, Nissim and Pinkas [FNP04]. However, their solution requires a costly secure-circuit evaluation since the intersection cardinality must remain private. Protocol 1, based on ideas from [AIR01, Lip03], is a conceptually simpler and more efficient alternative, especially when security either in the malicious model is required or the protocol is used in the context of itemset counting as later in Protocol 3. Here, we explicitly assume that the plaintext length is at least  $\text{len}(n)$  bits, where  $n \geq |\mathcal{Q} \cup \mathcal{S}|$  is the *a priori* fixed domain size. This assumption is always true in practice.

**Theorem 1.** *Let  $\Pi$  be a  $(\tau, \varepsilon)$  IND-CPA secure homomorphic cryptosystem and let  $n$  be smaller than any prime divisor of  $|\mathcal{P}|$ . Then Protocol 1 is  $(\tau - \mathcal{O}(n), n\varepsilon)$ -client-private and 0-server-private in the semi-honest model. Protocol 1 is correct with probability  $1 - |\mathcal{P}|^{-1}$ .*

*Proof.* First,  $\mathcal{Q} \subseteq \mathcal{S}$  iff  $w := \sum_{i:\mathcal{S}[i]=0} \mathcal{Q}[i] = 0$ . Therefore, homomorphic properties of  $\Pi$  assure that  $c$  is a random encryption of zero, if  $\mathcal{Q} \subseteq \mathcal{S}$ . If  $\mathcal{Q} \not\subseteq \mathcal{S}$ , then  $w \leq |\mathcal{Q}| \leq n$  is not a divisor of  $|\mathcal{P}|$  and thus  $c$  is a random encryption of a random plaintext. Consequently, the probability that  $\mathcal{Q} \not\subseteq \mathcal{S}$  if  $c$  is an encryption of zero is  $|\mathcal{P}|^{-1}$ . Computational client-privacy follows directly from the IND-CPA security of  $\Pi$ . As Server sees only  $n$  ciphertexts, any adversary  $A$  that can distinguish two vectors of ciphertexts can be used for distinguishing only two ciphertexts. The corresponding hybrid argument is fairly standard. Server-privacy is guaranteed as the second message depends only on whether  $\mathcal{Q} \subseteq \mathcal{S}$  or not.  $\square$

*Security in the malicious model.* A standard way to make the described protocol private in the malicious model is to let Client to prove the correctness of her actions; that means proving that (a)  $\text{pk}$  is a valid public key and that (b) every  $c_i$  encrypts either 0 or 1. This can be done by using (non-interactive) zero-knowledge or non-interactive zero-knowledge proofs of knowledge.

## 4 Exact Private Itemset Support Counting Protocols

Let  $\mathcal{Q} \subseteq [n]$  be Client's query,  $\mathcal{D}$  be the database and  $m$  be the number of the rows in the database; that is,  $\mathcal{D}[i] \subseteq [n]$  for  $i \in [m]$ . More precisely, we treat  $\mathcal{Q}$  as a binary  $n$ -tuple corresponding to the characteristic function of Client's input and  $\mathcal{D}$  as an  $m \times n$  binary matrix. Recall that in a PISC protocol, Client has to compute, in a privacy-preserving manner, the value  $\text{supp}_{\mathcal{D}}(\mathcal{Q}) := |\{i : \mathcal{Q} \subseteq \mathcal{D}[i]\}|$ .

### 4.1 Relation Between PISC and CPIR

We first show that there are tight reductions between oblivious transfer and PISC protocols even if  $n$  is relatively small. For precise quantification, denote by  $C_{\text{CPIR}}(s, \ell)$  the communication of a  $\binom{s}{1}$ -computationally private information retrieval protocol CPIR on  $\ell$ -bit strings. Similarly, let us denote by  $C_{\text{PISC}}(m, n)$  the communication of an  $(m \times n)$ -PISC protocol PISC.

**Theorem 2.** (a) Let CPIR be a  $\binom{2}{1}$ -computationally private information retrieval protocol on  $\ell$ -bit strings. Assume that  $\text{len}(m) \leq \ell$ . Then there exists a client-private  $(m \times n)$ -PISC protocol CPIR-PISC with communication  $C_{\text{CPIR}}(2^n, \ell)$ . Server has to pre-compute and store a table of  $2^n \cdot \text{len}(m)$  bits; this table can be computed in time  $\Theta(2^n \cdot m)$  ignoring logarithmically-small multiplicands.

(b) Let PISC be a client-private  $(2^n \times n)$ -PISC protocol. Then there exists a  $\binom{2}{1}$ -CPIR protocol PISC-CPIR on 1-bit strings with communication  $C_{\text{PISC}}(2^n, n)$ . Server has to pre-compute and store a table of  $\leq 2^n \cdot n$  bits; this table can be computed in time  $\Theta(2^{2n} \cdot 2^n)$  ignoring logarithmically-small multiplicands.

*Proof.* (a) Server computes off-line the support of all  $2^n$  itemsets in the database  $\mathcal{D}$ , and stores them in a new database  $\mathcal{D}'$ . Note that the database  $\mathcal{D}'$  contains  $2^n$  elements, each  $\text{len}(m)$  bits. After that, Client and Server use the  $\binom{2}{1}$ -CPIR protocol to retrieve the  $\mathcal{Q}$ th element of  $\mathcal{D}'$ . Clearly, Client learns the support of  $\mathcal{Q}$ , while Server obtains no new knowledge. If we use an oblivious transfer protocol instead of a CPIR protocol, then we get a server-private version of the CPIR-PISC protocol,

(b) Let  $\mathcal{S} = \mathcal{S}[1] \dots \mathcal{S}[2^n - 1]$  be Server's  $(2^n - 1)$ -bit input, and let  $i$  be Client's query in the  $\binom{2}{1}$ -CPIR protocol. We construct a specific  $2^n \times n$  binary database  $\mathcal{D}$  such that itemset supports in it encode  $\mathcal{S}$ . More precisely, let  $\chi(a) := (a_1, \dots, a_n)$  be a Boolean vector corresponding to the binary representation of  $a$ , that is,  $a = \sum 2^{j-1} a_j$ . The next algorithm builds a database  $\mathcal{D}$  such that  $\text{supp}_{\mathcal{D}}(\chi(a)) \equiv \mathcal{S}[a] \pmod{2}$  for every  $a \in \{1, \dots, 2^n - 1\}$ :

1. Initialise  $\mathcal{D}$  as a  $2^n \times n$  all-zero matrix.
2. For  $w = n$  downto 1 do

For all  $a$  s.t. the vector  $(a_1, \dots, a_n) \in \mathbb{Z}_2^n$  has Hamming weight  $w$  do

(a) Set  $v \leftarrow \text{supp}_{\mathcal{D}}(a_1, \dots, a_n)$ .

(b) If  $v \not\equiv \mathcal{S}[a] \pmod{2}$

then replace the first all-zero row of  $\mathcal{D}$  with  $(a_1, \dots, a_n)$ .

Since this algorithm considers itemsets in the order of decreasing cardinality, subsequent changes do not alter the already computed supports; thus, at the end all bits of  $\mathcal{S}$  are correctly encoded. Moreover, the number of replaced rows is not greater than  $2^n - 1$  and thus step 2b never fails to find an all-zero row. It is straightforward to derive the complexity bounds for this step.

Let  $\mathcal{D}$  be the final version of this database. Now, when Client wants to make a CPIR query  $i$ , he instead forms the corresponding PISC query  $\mathcal{Q} := \chi(i)$  and obtains  $\text{supp}_{\mathcal{D}}(\mathcal{Q})$  by executing PISC. Then, he computes  $\mathcal{S}[i] \leftarrow \text{supp}_{\mathcal{D}}(\mathcal{Q}) \bmod 2$ . Clearly, the client-privacy of PISC-CPIR follows directly from the client-privacy of the original PISC protocol.  $\square$

By using similar but more complicated techniques, one can directly construct an oblivious transfer protocol based on a PISC protocol. In this case, the number of rows of the database  $\mathcal{D}$  is still polynomial w.r.t. the number of encoded bits.

**Corollary 1.** *Assume that the Decisional Composite Residuosity Problem is hard. Assume that  $n = \text{polylog}(m)$ . There exists a private  $(m \times n)$ -PISC protocol CPIR-PISC with communication  $\Theta(n^2 \cdot \log_2 |\mathcal{P}| + n \cdot \text{len}(m))$  and Server's online work  $\Theta(2^n \cdot m)$ .*

The use of a very recent  $\binom{m}{1}$ -CPIR protocol by Gentry and Ramzan [GR05] would result in communication  $\Theta(n + \text{len}(m))$ .

As the communication complexity of non-private itemset support count is roughly  $n + \text{len}(m)$ , Corollary 1 provides an almost communication-optimal solution. On the other hand, Thm. 2 indicates that any PISC protocol with optimal communication  $\mathcal{O}(n + \log m)$  gives a rise to a  $\binom{s}{1}$ -CPIR protocol with communication  $\mathcal{O}(\log s)$ . Moreover, the known lower and upper bounds on the CPIR protocols can be used to get lower and upper bounds for the  $(m \times \text{polylog}(m))$ -PISC protocols. For example, given a trapdoor permutation, there exists an  $(m \times \text{polylog}(m))$ -PISC protocol with communication  $m - o(m)$ . On the other hand, an  $(m \times \text{polylog}(m))$ -PISC protocol with communication  $m - o(m)$  implies the existence of one-way functions.

## 4.2 Oblivious Keyword Search-Based PISC

As a serious drawback, note that CPIR-PISC is practical only for small values of  $n$ , e.g., when  $n \leq 20$ , as the pre-computation step becomes quickly infeasible. The same applies for the CPIR step, as Server's workload is at least linear in the size of database in all CPIR protocols.

However, in specific settings the online complexity of CPIR can be drastically reduced. The first efficient protocol for oblivious keyword search was proposed by Ogata and Kurosawa [OK04]; their approach was extended in [FIPR05]. In these two protocols, during the first phase Server transfers the whole database, in an encrypted form, to Client. In the second phase, Client and Server invoke an oblivious pseudo-random function evaluation protocol. As the result, Client obtains a secret key that allows her to decrypt one database element. Though the initial communication of the protocol is linear in the database size, the second phase has poly-logarithmic communication and computation in the database size. Such a protocol is especially appealing if the second phase are repeated many times as it is commonly done in data-mining algorithms.

PRIVATE INPUT: Client has a query  $\mathcal{Q}$  and Server has a database  $\mathcal{D}$ .

PRIVATE OUTPUT: Client learns  $\text{supp}_{\mathcal{D}}(\mathcal{Q})$  if  $\text{freq}_{\mathcal{D}}(\mathcal{Q}) \geq \sigma$ .

#### Setup Phase

Server runs a frequent itemset mining algorithm that outputs  $\mathcal{F} = \{X : \text{freq}_{\mathcal{D}}(X) \geq \sigma\}$ .

Server chooses a secret key  $\text{sk}$  and for all  $X \in \mathcal{F}$  computes  $E_i \leftarrow \text{ObPrf}_{\text{sk}}(\text{code}(X)) \oplus (0^\ell || \text{supp}_{\mathcal{D}}(X))$ .

Send list  $\{E_i\}$ , in a randomly permuted order, to Client.

#### Interactive Phase

Client and Server invoke  $\Pi_{\text{ObPrf}}$ . At the end Client obtains  $\text{mask} \leftarrow \text{ObPrf}_{\text{sk}}(\text{code}(X))$ .

#### Post-processing by Client

If there is an  $E_i$  such that  $E_i \oplus \text{mask} = (0^\ell || c)$  then output  $c$ ;

else decide that  $\text{supp}_{\mathcal{D}}(\mathcal{Q}) < \sigma$ .

### Protocol 2. Protocol for PISC based on oblivious pseudo-random function evaluation

Thm. 2 can be used to transform an oblivious keyword search protocol to a PISC protocol. If one is interested in the frequencies of all different supports, then the resulting protocol is not really practical since the transferred PISC database must have  $2^n$  elements. However, in data-mining applications, Client is often only interested in the supports of frequent itemsets. In such a case, Server can first run any conventional frequent itemset mining algorithm on the database using an appropriate minimum frequency threshold  $\sigma$ , and then encrypt supports of the obtained  $\sigma$ -frequent itemsets. In practice, the minimum frequency threshold  $\sigma$  is chosen to be as low as possible, so that the mining is still feasible, to get a good approximation of the supports of all itemsets.

Protocol 2 combines this idea with oblivious keyword search. This is relatively straightforward, but we have included the protocol for the sake of completeness. To read it, first recall that a two-argument pseudo-random function  $\text{ObPrf}$  is an OPRF, if it can be obliviously evaluated by Client and Server [FIPR05]. In other words, there exist a secure protocol  $\Pi_{\text{ObPrf}}$  such that after executing it on Client's private input  $x$  and Server's private input  $\text{sk}$ , Client learns  $\text{ObPrf}_{\text{sk}}(x)$ , while Server learns nothing. Second, we assume that each itemset  $\mathcal{Q}$  has a short unique code  $\text{code}(\mathcal{Q})$ , this can be a cryptographic hash of  $\mathcal{Q}$ .

**Theorem 3.** *Let  $\text{ObPrf}$  be  $(\tau, \varepsilon'_1)$  secure pseudo-random function with appropriate domain and range. Let the protocol  $\Pi_{\text{ObPrf}}(\tau, \varepsilon_1)$  client-private and  $(\tau, \varepsilon'_2)$  server-private. Then Protocol 2 is  $(\tau, \varepsilon_1)$ -client-private and  $(\tau - \mathcal{O}(1), \varepsilon'_1 + \varepsilon'_2)$ -server-private PISC protocol. Protocol 2 yields an incorrect end-result with probability  $2^{-\ell} \cdot |\mathcal{F}|$ . The interactive phase can be repeated over the same initially transformed encrypted database with a linear drop in concrete security.*

*Proof (Sketch).* We do not provide a complete proof, see [FIPR05] for details. Client-privacy and correctness are evident. Server-privacy follows from the next hybrid argument. First, consider a protocol  $\Pi_1$ , where  $\Pi_{\text{ObPrf}}$  is substituted with its ideal implementation. Let  $\Pi_2$  be the protocol, where also  $\text{ObPrf}$  is substituted with a random function. It is clear that  $\Pi_2$  is 0-server-private. The claim follows, since the protocols  $\Pi_2$  and  $\Pi_1$  are computationally  $\varepsilon'_1$ -close and  $\Pi_1$  and Protocol 2 are computationally  $\varepsilon'_2$ -close.  $\square$

Ogata and Kurosawa used the RSA blinded signature to construct an ObPrf, i.e., there  $\text{ObPrf}_{\text{sk}}(x) = \text{Prf}(\text{BlindSign}_{\text{sk}}(x))$  for a pseudo-random function Prf. However, the Ogata-Kurosawa protocol is *a priori* secure only in the random-oracle model. On the other hand, their protocol has only two messages and Server's actions are verifiable. Indeed, any two-message server-verifiable ObPrf can be converted to a blind signature scheme. Thus, the Ogata-Kurosawa construction is optimal in that sense. Freedman et al [FIPR05] proposed an alternative OPRF construction that is secure in the standard model under the Decisional Diffie-Hellman assumption. Unfortunately, their two-round solution is not *a priori* server-verifiable; that is, Client cannot test whether the obtained value is equal to  $\text{ObPrf}_{\text{sk}}(x)$ . Therefore, a malicious Server can mount undetectable denial of service attacks by deviating from  $\Pi_{\text{ObPrf}}$ .

If the OPRF is verifiable and the Setup phase is done correctly, then the whole protocol is verifiable; this is since Server cannot change the committed values  $E_i$ . As the Setup phase is relatively short and well-specified, its correctness can be guaranteed by non-cryptographic methods. The later does not hold for query-transfer phase as queries can arrive during a long time period.

Generally, Protocol 2 is well suited for static databases, where Server has to run frequent itemsets mining algorithm rarely, say once in a month. Otherwise, the large initial complexity over-weights its possible advantages.

### 4.3 On-Line Computation with Subset Inclusion

As stated before, the pre-computation cost of CPIR-PISC protocol is too large even in the case of the databases of moderate size. Limiting the answers only to frequent itemsets, like in Sect. 4.2, extends the applicability of CPIR-PISC to larger databases but limits the possible queries. To answer support queries also in large databases, we give Protocol 3. In this protocol, Server does not perform any pre-computation. Instead, when Server gets Client's query as an encrypted binary vector  $(c_1, \dots, c_n)$ , he runs a private subset inclusion test (a version of Protocol 1 that is secure in the semi-honest model) for every row of  $\mathcal{D}$ , and finally returns the replies in a randomised order. As a result, Client receives  $D_{\text{sk}}(C_i) = 0$  for every row where  $\mathcal{Q}$  was present and  $D_{\text{sk}}(C_i)$  is random element of  $\mathcal{P}$  for every row where  $\mathcal{Q}$  was not present. After that, she decrypts the results and then counts the number of nonzero values. Together with Protocol 1, the communication of this protocol (in the semi-honest model) is  $(n + 1 + m) \cdot \text{len}(|\mathcal{P}|)$  bits. (Note that here we implicitly need that  $|\mathcal{P}| > n$ .) No off-line work is done; Client performs  $\Theta(n + m)$  and Server does  $\Theta(w(\mathcal{D}) + m)$  units of online computation, where  $w(\mathcal{D})$  denotes the number of 1-s in the whole database. Again, in the data-mining scenarios, the database is usually very sparse and therefore  $w(\mathcal{D})$  is small.

**Theorem 4.** *Let  $\Pi$  be  $(\tau, \varepsilon)$  IND-CPA secure homomorphic cryptosystem. Then Protocol 3 is  $(\tau - \mathcal{O}(n), n\varepsilon)$  client-private and 0-server-private in the semi-honest model.*

*Proof.* As Server sees only  $n$  encryptions, client-privacy follows from standard hybrid argument: any adversary  $A$  that can distinguish two vectors of ciphertexts can be used for distinguishing only two ciphertexts. Simulation of Client's view is straightforward, the simulator must send  $\text{supp}_{\mathcal{D}}(\mathcal{Q})$  encryptions of 0's and  $m - \text{supp}_{\mathcal{D}}(\mathcal{Q})$  encryptions of random elements.  $\square$

PRIVATE INPUT: Client has a set  $\mathcal{Q}$  and Server has a database  $\mathcal{D}$ .

PRIVATE OUTPUT: Client knows  $\text{count} = \text{supp}_{\mathcal{D}}(\mathcal{Q})$

**Message 1, Client** Generate  $(\text{sk}, \text{pk}) \leftarrow G$  and send  $\text{pk}$  to Server.

For any  $i \in [n]$ , send  $c_i \leftarrow E_{\text{pk}}(\mathcal{Q}[i]; r_i)$  with  $r_i \leftarrow_r \mathcal{R}$  to Server.

**Message 2, Server** Generate a random permutation  $\pi : [m] \mapsto [m]$ .

Set  $d \leftarrow \prod_{i=1}^n c_i$ .

For every transaction  $j \in [m]$

Draw  $s_j \leftarrow_r \mathcal{P}$  and  $r'_j \leftarrow_r \mathcal{R}$  uniformly at random.

Send  $C_j \leftarrow (d / \prod_{i: \mathcal{D}[\pi(j), i]=1} c_i)^{s_j} \cdot E_{\text{pk}}(0; r'_j)$  to Client.

**Post-processing by Client**

Set  $\text{count} \leftarrow 0$ .

For every row  $j \in [m]$

If  $D_{\text{sk}}(C_j) = 0$  then  $\text{count} \leftarrow \text{count} + 1$ .

Return  $\text{count}$ .

**Protocol 3.** Protocol for PISC, based on the private inclusion test

Security against malicious Clients is achieved by letting Client to prove in zero-knowledge, for every  $i \in [n]$ , that  $c_i$  is an encryption of either 0 or 1. This is usually feasible since in reality,  $n \leq 100\,000$ .

## 5 Imprecise Private Itemset Counting Protocols

In practice, it is not usually necessary to give exact supports but accurate approximations. Sometimes it is even sufficient to know whether a set is frequent or not. In this section, we consider two approaches to approximate frequency queries. First, we study protocols to decide whether a given itemset is frequent or not. That gives rise to deterministic support approximation techniques. Second, we show how random sampling can be used together with Protocol 3 to obtain support approximations with quality guarantees.

### 5.1 Private Frequent Itemset Detection

The simplest approximation of the support of a frequent itemset is to tell whether or not the itemset is frequent. At the end of a *private frequent itemset detection (PFID) protocol*, Client learns whether  $\text{freq}_{\mathcal{D}}(\mathcal{Q}) \geq \sigma$ , and nothing else. PFID is a common subtask in pattern discovery [MT97, GKM<sup>+</sup>03]. Moreover, PFID can be used as sub-protocol in different approximate PISC protocols.

One straightforward solution is to use Prot. 2 on a database where one stores some fixed integer instead of the cardinality of the support. However, this does not decrease communication or computation significantly.

A more interesting alternative is to modify the database so that it contains only maximal frequent sets, i.e., frequent sets that are not subsets of other frequent sets. That is, every maximal frequent set is added to a new database  $\mathcal{D}'$  as a transaction. Afterwards,



Client and Sever execute a PISC protocol on  $\mathcal{D}'$ . If  $\text{supp}_{\mathcal{D}'}(\mathcal{Q}) \geq 0$ , then  $\mathcal{Q}$  is frequent in the original database. Since the number of maximal itemsets can be exponentially smaller than the number of frequent itemsets, this idea might give us a significant win in the practice. Note that instead of just the support of  $\mathcal{Q}$ , the resulting protocol also outputs both the number  $\ell$  of maximal itemsets and the number  $k$  of maximal itemsets that contain  $\mathcal{Q}$ . This additional information is sometimes desired in data mining. Even if undesired, however, such a leak is not necessarily very dangerous for the privacy of the database, since in practice there are always many different alternatives that could be the collection of the  $\ell$  maximal itemsets,  $k$  of them containing the itemset  $\mathcal{Q}$ . Namely, any anti-chain (i.e., a collection  $\mathcal{A}$  such that  $(X \subseteq Y \vee Y \subseteq X) \Rightarrow X = Y$  for all  $X, Y \in \mathcal{A}$ ) of  $\ell$  itemsets on the attributes of the database such that  $k$  of the itemsets contain  $\mathcal{Q}$  and  $\ell - k$  do not contain  $\mathcal{Q}$  could be that collection of maximal itemsets.

To show that the number of such possible collections of maximal itemsets is large, we give a rough lower bound in the case the query contains less than  $n/2$  elements and there is no prior information on the database content. Let  $\mathcal{M}$  be a collection of all itemsets of cardinality  $\lfloor n/2 \rfloor$ . Let  $c_{\mathcal{Q}}$  be the cardinality of  $\{X \in \mathcal{M} : \mathcal{Q} \subseteq X\}$  and  $c$  cardinality of  $\mathcal{M}$ , i.e.  $c = \binom{n}{\lfloor n/2 \rfloor}$ . Now, there is exactly  $\binom{c_{\mathcal{Q}}}{k}$  ways to choose maximal sets from  $\mathcal{M}$  containing  $\mathcal{Q}$ , and  $\binom{c - c_{\mathcal{Q}}}{\ell - k}$  ways to choose other  $\ell - m$  elements that cannot contain  $\mathcal{Q}$ . Therefore, the lower bound on consistent configurations is huge,

$$\binom{c_{\mathcal{Q}}}{k} \binom{c - c_{\mathcal{Q}}}{\ell - k} = \binom{\binom{n - |\mathcal{Q}|}{\lfloor n/2 \rfloor - |\mathcal{Q}|}}{k} \left( \binom{n}{\lfloor n/2 \rfloor} - \binom{n - |\mathcal{Q}|}{\lfloor n/2 \rfloor - |\mathcal{Q}|} \right).$$

Furthermore, the exact numbers of maximal itemsets can be hidden by adding adding some subsets of the maximal itemsets to the database. This does not change the outcome of the protocol since if an itemset is contained in some subset of a maximal itemset, then it is obviously contained also in the maximal itemset itself. Note that also the supports of the itemsets leak information about other itemsets and the underlying database since each acquired support further restricts the collection of databases compatible with the known supports [Mie03].

A PFID protocol can be used to answer exact and approximate support queries. Let  $0 < \sigma_1 < \dots < \sigma_k \leq 1$  be the minimum frequency thresholds for which the frequent itemset detection problem can be solved. Then Client can find out to which of the intervals  $(0, m\sigma_1], \dots, (m\sigma_k, m]$  the support of  $\mathcal{Q}$  belongs. Note that there are at most  $m$  different possible supports and in practice the number is often strictly smaller than that.

## 5.2 Sampling

A randomly chosen subset of the transactions provides usually a very good summary of the database. This is true also in the case of frequency estimation. Recall that the frequency of an itemset in a transaction database is the fraction of the transactions containing that itemset. Thus, the frequency of an itemset can be estimated from a set of randomly chosen transactions. Furthermore, the relative and absolute errors of such an estimation can be bounded by the Chernoff bound and the Hoeffding bound, respectively. (Note that Protocol 3 can be applied as well to a randomly chosen subset of transactions of the database as to the database itself.)

Let us assume for a moment that we know that the frequency of the itemset is at least  $\sigma$ . In that case the Chernoff bound gives us the following bound for the relative error  $\varepsilon$ :

**Theorem 5.** *Let  $\mathcal{S}$  be a random  $k$ -row sample of an  $n$ -row database  $\mathcal{D}$  with replacement. Then  $\Pr[(1 - \varepsilon) \text{freq}_{\mathcal{D}}(\mathcal{Q}) \leq \text{freq}_{\mathcal{S}}(\mathcal{Q}) \leq (1 + \varepsilon) \text{freq}_{\mathcal{D}}(\mathcal{Q})] \leq 2 \exp(-\varepsilon^2 k \sigma / 4)$ , when the itemset is frequent, i.e.  $\text{freq}_{\mathcal{D}}(\mathcal{Q}) \geq \sigma$ .*

*Proof.* Let  $I_{\mathcal{Q}}(\mathcal{S}[i])$  be  $I_{\mathcal{Q}}$  be an indicator variable, i.e.  $I_{\mathcal{Q}}(\mathcal{S}[i]) = 1$  if  $\mathcal{Q} \subseteq \mathcal{S}[i]$  and 0 otherwise. Then the frequency estimator  $\text{freq}_{\mathcal{S}}(\mathcal{Q}) = \sum_{i=1}^k I_{\mathcal{Q}}(\mathcal{S}[i]) / k$  is a random variable with the expectation  $\mathbf{E}(\text{freq}_{\mathcal{S}}(\mathcal{Q})) = \text{freq}_{\mathcal{D}}(\mathcal{Q})$ . As  $\text{freq}_{\mathcal{S}}(\mathcal{Q})$  is a sum of  $k$  i.i.d. random zero-one variables and  $\text{freq}_{\mathcal{D}}(\mathcal{Q}) \geq \sigma$  by assumption, we can apply the Chernoff bound and underestimate  $\text{freq}_{\mathcal{D}}(\mathcal{Q})$  by  $\sigma$ . This proves the claim.  $\square$

Theorem 5 allows us to estimate the sufficient number of transactions to bound the relative error  $\varepsilon$  and the failure probability  $\delta$ :

**Corollary 2.** *To guarantee that the relative error is at most  $\varepsilon$  and the failure probability is at most  $\delta$ , it is sufficient to randomly choose  $k \geq 4(\ln 2 / \delta) / (\varepsilon^2 \sigma)$  transactions from the database.*

Moreover, if we are interested in bounding the absolute error, the number of rows sufficient to guarantee a certain error bound is even smaller without assuming anything about the frequencies:

**Theorem 6.** *Let  $\mathcal{S}$  be a random  $k$ -row sample of an  $n$ -row database  $\mathcal{D}$  with replacement. Then  $\Pr[|\text{freq}_{\mathcal{S}}(\mathcal{Q}) - \text{freq}_{\mathcal{D}}(\mathcal{Q})| \geq \varepsilon] \leq 2 \exp(-2\varepsilon^2 k)$ .*

*Proof.* As  $\text{freq}_{\mathcal{S}}(\mathcal{Q})$  is a sum of  $k$  i.i.d. random zero-one variables, we can apply the Hoeffding bound which proves the claim.  $\square$

The number of transaction sufficient in this case is as follows:

**Corollary 3.** *To guarantee that the relative error is at most  $\varepsilon$  and the failure probability is at most  $\delta$ , it is sufficient to randomly choose  $k \geq (\ln 2 / \delta) / (2\varepsilon^2)$  transactions from the database.*

For example, with failure probability  $10^{-3}$  and absolute error  $1/100$ , it is sufficient to have 38 500 rows in the sample and thus PSI-PISC protocol is efficient enough. Hence, the sampling technique provides an approximation protocol such that the computation and communication complexity are independent from database size. The complexity depends only on the desired approximation error  $\varepsilon$  and the failure probability  $\delta$ .

The approximation error bounds given above can be used also to obtain approximations for the frequent itemset detection, i.e., *property testers* for itemsets being frequent. More specifically, we can use the above bounds to decide correctly with high probability whether the itemset is frequent or not when the correct frequency of the itemset is above or below the minimum frequency threshold  $\sigma$  at least by error  $\varepsilon$ ; if the correct frequency is within the error  $\varepsilon$  from the minimum frequency threshold  $\sigma$ , we might answer incorrectly with non-negligible probability.

Chernoff and Hoeffding bounds are quite tight when estimating the frequency of a single itemset from one sample databases, i.e. we re-sample the database before each

query. This is not the case when several frequencies are estimated, i.e., when Server generates a single database by sampling the transactions to answer all or many Client's queries. In this case, Server might even verify the approximation precision of all frequent itemsets  $\mathcal{F}$  (or some other collection of itemsets of interest).

The straightforward generalisation of the Hoeffding bound to a collection  $\mathcal{F}$  of itemsets with maximum absolute error guarantee  $\varepsilon$  for an arbitrary sequence of frequency queries to  $\mathcal{F}$  yields to sample complexity of  $(\ln 2 |\mathcal{F}| / \delta) / (2\varepsilon^2)$  [Toi96]. However, this is a worst-case bound. In practice, transaction databases are not as difficult to sample as possible but have a lot structure. One important measure of complexity of a transaction database is its Vapnik-Chervonenkis dimension (VC-dimension) w.r.t. the itemsets whose frequencies want to be able to estimate accurately. If the VC-dimension is  $k$ , then the number of transactions is sufficient to guarantee maximum absolute error  $\varepsilon$  with probability  $1 - \delta$  is  $\mathcal{O}(k \ln 1 / ((\delta\varepsilon) / \varepsilon^2))$ .

For example, the VC-dimension of the database w.r.t. frequent itemsets can be bounded above by  $\log |\mathcal{C}|$ , where  $\mathcal{C}$  is the collection of closed frequent itemsets in the database [Mie04]. (An itemset is closed in collection  $\mathcal{F}$  if its support is strictly higher than the largest support of its supersets.) Using this upper bound for the Vapnik-Chervonenkis dimension, it is possible to show that a sample of  $\mathcal{O}(\ln |\mathcal{C}| \ln 1 / ((\delta\varepsilon) / \varepsilon^2))$  transactions suffices to guarantee with failure probability  $\delta$  that the maximum absolute error is at most  $\varepsilon$ . As the number of closed frequent itemsets can be exponentially smaller than the number of frequent itemsets, the VC-based sample bound can be  $\mathcal{O}(\ln \ln |\mathcal{F}| \ln 1 / ((\delta\varepsilon) / \varepsilon^2))$  at smallest. In practice, one can compute both the Hoeffding bound and the VC-bound, and take the minimum of them. Also, if the collection of the itemsets for which the approximation guarantees are required is small enough, then one can maintain the frequency estimates for all itemsets of interest and stop the sampling immediately when the frequency estimates are sufficiently accurate.

**Acknowledgements.** We would like to thank Bart Goethals for proposing this problem and for many useful discussions and comments, and Aggelos Kiayias and Antonina Mitrofanova for comments. The first author was partially supported by the Finnish Academy of Sciences. The second author was partially supported by the Estonian Science Foundation, grant 6096.

## References

- [AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag.
- [AMS<sup>+</sup>96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast Discovery of Association Rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.
- [BGZ04] Roberto J. Bayardo Jr., Bart Goethals, and Mohammed Javeed Zaki, editors. *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, volume 126 of *CEUR Workshop Proceedings*, Brighton, UK, November 1, 2004.

- [CGN97] Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. Technical Report TR CS0917, Department of Computer Science, Technion, 1997.
- [FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword Search and Oblivious Pseudorandom Functions. In Joe Kilian, editor, *The Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324, Cambridge, MA, USA, February 10–12, 2005. Springer Verlag.
- [FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient Private Matching and Set Intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag.
- [GKM<sup>+</sup>03] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharma. Discovering All Most Specific Sentences. *ACM Transactions on Database Systems*, 28(2):140–174, 2003.
- [GR05] Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In Luis Caires, Guisepppe F. Italiano, Luis Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815, Lisboa, Portugal, 2005. Springer-Verlag.
- [Lip03] Helger Lipmaa. Verifiable Homomorphic Oblivious Transfer and Private Equality Test. In Chi Sung Lai, editor, *Advances on Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 416–433, Taipei, Taiwan, November 30–December 4, 2003. Springer-Verlag.
- [Lip05] Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In Jianying Zhou and Javier Lopez, editors, *The 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages ?–?, Singapore, September 20–23, 2005. Springer-Verlag. To appear.
- [Mie03] Taneli Mielikäinen. On Inverse Frequent Set Mining. In *2nd Workshop on Privacy Preserving Data Mining (PPDM)*, pages 18–23, Melbourne, Florida, USA, November 19, 2003. IEEE Computer Society.
- [Mie04] Taneli Mielikäinen. Separating Structure from Interestingness. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004*, volume 3056 of *Lecture Notes in Computer Science*, pages 476–485, Sydney, Australia, May 24–28, 2004. Springer.
- [MT97] Heikki Mannila and Hannu Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
- [OK04] Wakaha Ogata and Kaoru Kurosawa. Oblivious Keyword Search. *Journal of Complexity*, 20(2–3):356–371, 2004.
- [Toi96] Hannu Toivonen. Sampling Large Databases for Association Rules. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases*, pages 134–145, Mumbai, India, September 3–6, 1996. Morgan Kaufmann.

# Visual Cryptographic Protocols Using the Trusted Initializer

Hidenori Kuwakado<sup>1</sup>, Masakatu Morii<sup>1</sup>, and Hatsukazu Tanaka<sup>2</sup>

<sup>1</sup> Kobe University,

1-1 Rokkodai Nada Kobe 657-8501, Japan

<sup>2</sup> Kobe Institute of Computing,

2-2-7 Kano Chuo Kobe 650-0001, Japan

**Abstract.** We show how to visualize an oblivious transfer and a commitment scheme with a method similar to that used in a visual secret sharing scheme. We call them a visual oblivious transfer and a visual commitment scheme, respectively. Data are images printed on transparencies, and the operation for obtaining information is only overlaying each of the transparencies over each other. Hence, it is easy for non-expert users to execute them. The visual oblivious transfer and the visual commitment scheme proposed in this paper are based on the trusted initializer model.

## 1 Introduction

The development of multi-party protocols is an active area of research. Due to many researchers, multi-party protocols that are seemingly impossible have been developed. Multi-party protocols require several fundamental tools based on the modern cryptography. Typical fundamental tools are a secret sharing scheme, an oblivious transfer, and a commitment scheme. There are many ways of constructing these tools, e.g., [1][2][3][4][5][6] for the secret sharing scheme, [7][8][9][10][11] for the oblivious transfer, and [12][8][9] for the oblivious transfer.

These tools require users to have cryptographic knowledge and to perform complicated operations. Such requirements probably cause non-expert users to hesitate to use these tools. However, Naor and Shamir [13] have shown the secret sharing scheme that does not require cryptographic knowledge and complicated operations. Their scheme is called a visual secret sharing scheme. It involves breaking up the secret image into images printed on transparencies, and the secret image can be visually recovered by placing the transparencies on top of each other.

In contrast, there is yet no way to construct the oblivious transfer and the commitment scheme without such requirements. We show how to visualize the oblivious transfer and the commitment scheme with a method similar to that used in the visual secret sharing scheme. We call them a visual oblivious transfer and a visual commitment scheme. The use of the visual oblivious transfer and the visual commitment scheme enables users to not only recover information but also check the validity of data by the human visual system, i.e., without

cryptographic knowledge and complicated operations. We construct the visual oblivious transfer and the visual commitment scheme using a trusted initializer who is active only at the setup of the protocol [14].

The visual secret sharing scheme suggests that executing the secret sharing scheme does not necessarily require a computer as a physical device. The difference in physical devices results in the difference in algebraic structures used for constructing the scheme. The algebraic structure of visual secret sharing schemes for a black and white image is a monoid, which is a set  $\{0, 1\}$  under the inclusive-OR operation. That for a color image is a lattice rather than the monoid [15]. The monoid and the lattice are weaker algebraic structures than a finite field and a finite group used in modern cryptography. It is significant to study the construction of cryptographic protocols under weak algebraic structures because a new physical device for executing cryptographic protocols will be developed in future. Our results imply that the oblivious transfer and the commitment scheme can be constructed over the monoid of  $\{0, 1\}$  of the weak algebraic structure.

We summarize notation. We let  $\oplus$ ,  $\vee$ , and  $\bar{\cdot}$  denote the exclusive-OR operator, the inclusive-OR operator, and the complement operator. We use these operators for not only computation of a bit but also that of a binary vector, that of a binary matrix, and that of a set. We define these computations by those of each component.

This paper is organized as follows. In Sect. 2 we describe the trusted initializer model and non-visual protocols. In Sect. 3 we show the visual oblivious transfer, analyze the security and the size of data, and give examples. In Sect. 4 we show the visual commitment scheme, analyze the security and the size of data, and give examples. In Sect. 5 we conclude this paper.

## 2 Protocols Based on the Trusted Initializer Model

### 2.1 Trusted Initializer

The trusted initializer model has been introduced by Rivest [14]. In this model there exists a privileged person (called a trusted initializer), and the privileged person initializes a protocol and after then becomes inactive. Specifically, the trusted initializer distributes data to users through private channels. Note that these channels are one-way, i.e., only from the trusted initializer to users. The step for distributing data is called the initializing step. After then, the trusted initializer does not participate in the protocol, and users perform the protocol using data distributed by the trusted initializer. The trusted initializer model enables us to construct unconditionally secure cryptographic protocols. In this paper, we discuss only protocols based on the trusted initializer model. We call the trusted initializer Ted, and call users Alice and Bob.

### 2.2 Oblivious Transfer

Oblivious transfers are classified into two types. The first-type oblivious transfer is as follows. When Alice sends information to Bob, he can obtain it with proba-

bility  $1/2$  and she cannot know whether he obtained it or not. The second-type oblivious transfer is as follows. When Alice sends two pieces of information to Bob, he can obtain only one of them and cannot obtain any information about another piece, and she cannot know which piece he obtained. Crépeau [16] has proved that both types of oblivious transfers are theoretically equivalent.

We show the first-type of a non-visual oblivious transfer since the second-type of a non-visual oblivious transfer has been given in [14]. For simplicity, suppose that Alice sends one-bit information to Bob.

Initializing step: Ted randomly chooses a two-dimensional vector  $\mathbf{v} = (v_0, v_1)$  where  $v_0, v_1 \in \{0, 1\}$ . After Ted chose a random bit  $c$ , Ted defines  $\mathbf{u}_c$  as  $\mathbf{u}_c = \mathbf{v}$  and determines  $\mathbf{u}_{\bar{c}}$  such that  $\mathbf{u}_{\bar{c}} \neq \mathbf{u}_c$  and  $\mathbf{u}_{\bar{c}} \oplus \mathbf{v} \neq (1, 1)$ . Since there are two candidates of  $\mathbf{u}_{\bar{c}}$ , Ted chooses one vector from them at random. Ted gives  $\{\mathbf{u}_0, \mathbf{u}_1\}$  and  $\mathbf{v}$  to Alice and Bob, respectively.

Transferring step: Let  $b \in \{0, 1\}$  be one-bit information that Alice wants to send to Bob. Alice computes a vector  $\mathbf{t}$  as follows:

$$\mathbf{t} = \begin{cases} (0, 0) \oplus \mathbf{u}_r, & \text{if } b = 0; \\ (1, 1) \oplus \mathbf{u}_r, & \text{otherwise.} \end{cases}$$

Alice sends  $\mathbf{t}$  to Bob. Bob computes a vector  $\mathbf{z}$  by  $\mathbf{z} = \mathbf{t} \oplus \mathbf{v}$ . Then, Bob finds the value of  $b$  as follows:

$$b = \begin{cases} 0, & \text{if } \mathbf{z} = (0, 0); \\ 1, & \text{if } \mathbf{z} = (1, 1); \\ \perp, & \text{otherwise,} \end{cases}$$

where ‘ $\perp$ ’ indicates that Bob cannot obtain any information about  $b$ .

By using a simple check it is not hard to verify that the above oblivious transfer works well. The size of data obtained by Bob, i.e., the size of  $\mathbf{z}$ , is the smallest because two bits is necessary for expressing ‘0’, ‘1’, and ‘ $\perp$ ’. In this sense the above oblivious transfer is optimal.

### 2.3 Commitment Scheme

A commitment scheme based on the trusted initializer must be designed in such a way that it satisfies the following properties. (i) Correctness: If Alice and Bob follow the scheme, Bob accepts  $x$  to which Alice committed. (ii) Concealment: If Alice and Bob follow the scheme, Bob cannot know any information about  $x$  before the revealing step. (iii) Cheating probability: Alice cannot succeed in cheating Bob with probability more than  $\epsilon$  ( $0 \leq \epsilon < 1$ ).

Blundo, Masucci, Stinson, and Wei [17] have proposed a commitment scheme (an AP scheme), which is a fixed version of Rivest’s commitment scheme [14]. The AP scheme is constructed over  $\text{GF}(p)$  where  $p$  is a prime. The AP scheme is unconditionally concealing, and its cheating probability is  $1/p$  [17].

### 3 Visual Oblivious Transfer

In Sect. 3 and Sect. 4, ‘0’ and ‘1’ mean a white (transparent) pixel and a black pixel printed on the transparency, respectively. The inclusive-OR operator means stacking transparencies. For example, “ $0 \vee 1 = 1$ ” means that stacking the white pixel on the black pixel results in the black pixel.

#### 3.1 Implementation for One-Bit Information

We show an implementation of the first-type visual oblivious transfer for one-bit information.

Initializing step: Ted chooses a four-dimensional binary vector  $\mathbf{v}$  with weight 2 at random where the weight is the number of nonzero components. After Ted chose a random bit  $c$ , he determines two four-dimensional binary vectors  $\mathbf{u}_i$  ( $i = 0, 1$ ) as follows. He defines  $\mathbf{u}_c$  by  $\mathbf{u}_c = \mathbf{v}$ , and determines  $\mathbf{u}_{\bar{c}}$  such that the weight of  $\mathbf{u}_{\bar{c}}$  is two and the weight of  $\mathbf{u}_{\bar{c}} \vee \mathbf{v}$  is three. Since such a vector is not unique, he chooses it from candidates at random. Ted sends the following  $\mathcal{U}, \mathcal{V}$  to Alice and Bob through their private channels, respectively.

$$\mathcal{U} = \{\mathbf{u}_0, \mathbf{u}_1\}, \quad \mathcal{V} = \{\mathbf{v}\}.$$

Transferring step: Let us consider that Alice sends one-bit information  $b$  ( $\in \{0, 1\}$ ) to Bob. After Alice chose a random bit  $r$ , she determines  $\mathbf{t}$  as follows:

$$\mathbf{t} = \begin{cases} \mathbf{u}_r, & \text{if } b = 0; \\ \overline{\mathbf{u}_r}, & \text{otherwise.} \end{cases}$$

Alice sends  $\mathcal{T} = \{\mathbf{t}\}$  to Bob. After Bob received it, he stacks  $\mathbf{t}$  on  $\mathbf{v}$ , i.e.,

$$\mathbf{z} = \mathbf{t} \vee \mathbf{v}.$$

If the weight of  $\mathbf{z}$  is two, one-bit information sent by Alice is 0, if the weight of  $\mathbf{z}$  is four, it is 1. Otherwise it is  $\perp$ , i.e., Bob does not obtain  $b$ .

Bob can visually recognize the result because their contrasts are different. Precisely, the contrast  $C$ , which is defined in [13], of each case is given as follows:

$$C = \begin{cases} 1/4, & \text{between } 0 \text{ and } \perp; \\ 1/4, & \text{between } 1 \text{ and } \perp; \\ 1/2, & \text{between } 0 \text{ and } 1. \end{cases}$$

We give an example of the above implementation. Suppose that Ted chose  $\mathbf{v} = (1, 1, 0, 0)$ . Let  $c = 1$ . It follows that  $\mathbf{u}_1 = \mathbf{v}$ . The set  $\widehat{\mathcal{U}}$  of candidates of  $\mathbf{u}_0$  is given by

$$\widehat{\mathcal{U}} = \{(0, 1, 0, 1), (0, 1, 1, 0), (1, 0, 0, 1), (1, 0, 1, 0)\}. \quad (1)$$



**Table 1.** Communication complexity in oblivious transfers [bits]

Step		Non-visual	Visual
Initializing	Alice	4 (3)	$8 (\log_2 6 + \log_2 4)$
	Bob	2 (2)	$4 (\log_2 6)$
Transferring		2 (2)	$4 (\log_2 6)$
Total complexity		8 (7)	$16 (3 \log_2 6 + \log_2 4)$

Suppose that Ted chose  $\mathbf{u}_0 = (0, 1, 0, 1)$ . After Ted randomly chose one vector from  $\widehat{\mathcal{U}}$  as  $\mathbf{u}_0$ , Ted sends  $\mathcal{U} = \{\mathbf{u}_0, \mathbf{u}_1\}$  and  $\mathcal{V} = \{\mathbf{v}\}$  to Alice and Bob, respectively. Suppose that Alice wants to send  $b = 0$ . After Alice chose  $r = 1$ , she sends  $\mathcal{T} = \{(1, 1, 0, 0)\}$  to Bob since  $\mathbf{t} = \mathbf{u}_1$ . Since Bob obtains  $\mathbf{z} = (1, 1, 0, 0)$ , he finds  $b = 0$ .

If Alice had chosen  $r = 0$ , Bob would not have obtained any information about  $b$ . Specifically, since  $\mathbf{t} = (0, 1, 0, 1)$ , it follows that  $\mathbf{z} = (1, 1, 0, 1)$ . We state the reason that Bob cannot obtain any information about  $b$  when  $c \neq r$ . When Alice wants to send  $b = 1$  and  $r = 0$ , she sends  $\mathbf{t} = \overline{\mathbf{u}_0} = (1, 0, 1, 0)$ . Notice that  $\overline{\mathbf{u}_0}$  is also an element of  $\widehat{\mathcal{U}}$ . We see from Eq. (1) that for any  $\mathbf{u} \in \widehat{\mathcal{U}}$  we have  $\overline{\mathbf{u}} \in \widehat{\mathcal{U}}$ . The following theorem formally guarantees that Bob cannot obtain any information about  $b$  when he finds  $\perp$ .

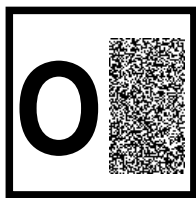
**Theorem 1.** *Let  $\mathbf{v}$  be a four-dimensional binary vector with weight 2. We denote by  $\widehat{\mathcal{U}}$  the set of four-dimensional binary vectors  $\mathbf{u}$  such that the weight of  $\mathbf{u}$  is two and the weight of  $\mathbf{u} \vee \mathbf{v}$  is three. Then, for any  $\mathbf{u} \in \widehat{\mathcal{U}}$  we have  $\overline{\mathbf{u}} \in \widehat{\mathcal{U}}$ .*

We next compare the above visual oblivious transfer to the non-visual oblivious transfer given in Sect. 2.2 in terms of the communication complexity. Table 1 shows the communication complexity of the oblivious transfers. The value in the bracket of Table 1 is the essential size of data. For example, in the initializing step of the non-visual oblivious transfer, Alice receives 4-bit data of  $\{\mathbf{u}_0, \mathbf{u}_1\}$ , but the data can be expressed with 3 bits because  $\mathbf{u}_0$  and  $\mathbf{u}_1$  are not independent. The smallest communication complexity of the transferring step is  $\log_2 3$  ( $\approx 1.58$ ) bits. Since  $\log_2 6 \approx 2.58$ , the communication complexity of the visual oblivious transfer is slightly larger than the smallest communication complexity. There might exist a more efficient visual oblivious transfer.

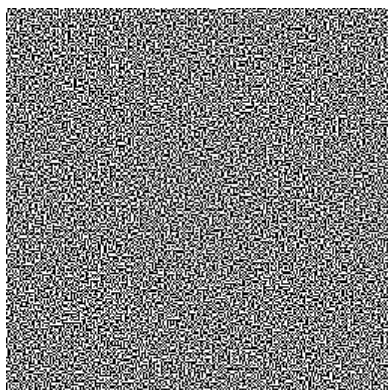
### 3.2 Application to the Image

We can construct the first-type visual oblivious transfer of multi-bit information  $\mathcal{B} = (b_1, b_2, \dots)$  by repeating that of one-bit information. We note that Bob obtains entire  $\mathcal{B}$  with probability  $1/2$ . It does not mean that Bob obtains each bit of  $\mathcal{B}$  with probability  $1/2$ . Hence, random bits  $c, r$  described in Sect. 3.1 are fixed for all  $b_j$  ( $j = 1, 2, \dots$ ).

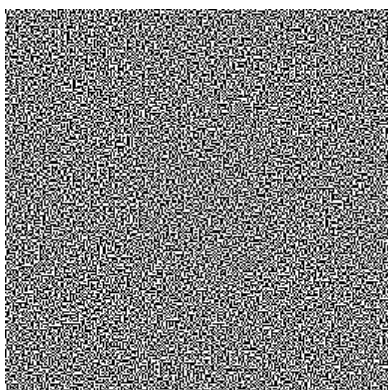
Using a black and white image  $\mathcal{B}$  of Fig. 1, we give an example of the visual oblivious transfer. Fig. 2 is an image  $\mathcal{V}$  of the set of  $\mathbf{v}$ . Fig. 3 and Fig. 4 are



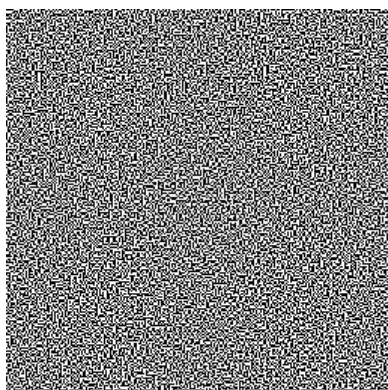
**Fig. 1.** Image  $B$



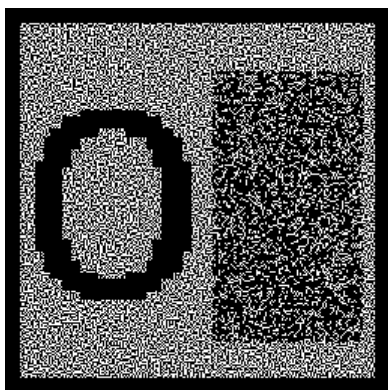
**Fig. 2.** Image  $\mathcal{V}$  from Ted to Bob



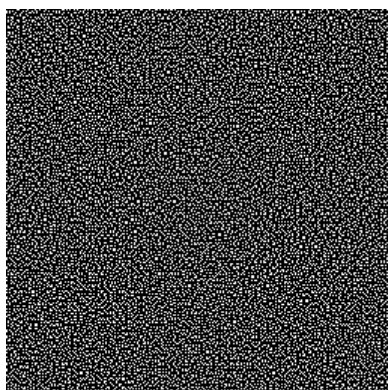
**Fig. 3.** Image  $\mathcal{T}$  when  $c = r$



**Fig. 4.** Image  $\mathcal{T}$  when  $c \neq r$



**Fig. 5.** Image  $\mathcal{Z} (= \mathcal{V} \vee \mathcal{T})$  when  $c = r$



**Fig. 6.** Image  $\mathcal{Z} (= \mathcal{V} \vee \mathcal{T})$  when  $c \neq r$

images  $\mathcal{T}$  of the set of  $\mathbf{t}$  when  $c = r$  and  $c \neq r$  respectively. Fig. 5 and Fig. 6 are images  $\mathcal{Z}$  of the set of  $\mathbf{z}$ , which Bob obtains by stacking the image  $\mathcal{T}$  on the image  $\mathcal{V}$ , when  $c = r$  and  $c \neq r$  respectively. Thus, if  $c = r$ , then Bob succeeds in recognizing  $\mathcal{B}$  visually, otherwise he fails. We note that Bob finds his failure because  $\mathcal{Z}$  is uniformly dark, not because no meaningful image is recovered. Comparing the right-hand area of Fig. 5 with Fig. 6, we find the difference in the variance of the number of white pixels of subpixels. The variance of the right-hand area of Fig. 5 is  $1/16$  whereas that of Fig. 6 is 0.

The above visual oblivious transfer uses 4 pixels for one-bit information, but the reviewer of this paper shows the visual oblivious transfer can be constructed by using 2 pixels for one-bit information in reviewer's report. Reviewer's visual oblivious transfer is useful for data involving much redundancy such as an image because one-bit information is not recognized precisely.

## 4 Visual Commitment Scheme

### 4.1 Implementation for One-Bit Information

We show an implementation of the visual commitment scheme for one-bit information.

Initializing step: Ted randomly chooses a matrix  $u$  from  $\widehat{\mathcal{U}}$ .

$$\widehat{\mathcal{U}} = \left\{ \begin{pmatrix} 00 \\ 11 \end{pmatrix}, \begin{pmatrix} 11 \\ 00 \end{pmatrix}, \begin{pmatrix} 10 \\ 10 \end{pmatrix}, \begin{pmatrix} 01 \\ 01 \end{pmatrix} \right\}$$

Ted randomly chooses one of two elements of '1' in  $u$ , and produces a matrix  $v$  such that the chosen element is replaced with '0' and the others are done with '1'. The resulting matrix  $v$  is an element of  $\widehat{\mathcal{V}}$ .

$$\widehat{\mathcal{V}} = \left\{ \begin{pmatrix} 11 \\ 01 \end{pmatrix}, \begin{pmatrix} 11 \\ 10 \end{pmatrix}, \begin{pmatrix} 01 \\ 11 \end{pmatrix}, \begin{pmatrix} 10 \\ 11 \end{pmatrix} \right\}$$

Ted sends  $u$  and  $v$  to Alice and Bob through their private channels.

Committing step: Let  $b (\in \{0, 1\})$  be one-bit information to which Alice wants to commit. If  $b = 0$ , then Alice transposes  $u$ , and permutes the rows and the columns of the transposed matrix at random. If  $b = 1$ , then Alice permutes the rows and the columns of  $u$  at random. The resulting matrix  $t$  is an element of  $\widehat{\mathcal{T}}$ . Alice sends  $t$  to Bob.

$$\widehat{\mathcal{T}} = \left\{ \begin{pmatrix} 00 \\ 11 \end{pmatrix}, \begin{pmatrix} 11 \\ 00 \end{pmatrix}, \begin{pmatrix} 10 \\ 10 \end{pmatrix}, \begin{pmatrix} 01 \\ 01 \end{pmatrix} \right\}$$

Revealing step: Alice sends  $u$  and  $b$  to Bob. Bob first stacks  $u$  on  $v$ , and checks that all the pixels are black. If not so, then Bob rejects  $b$ . Bob next stacks  $u$  on  $t$ , i.e.,  $z = u \vee t$ , and counts the number of white pixels of  $z$ . If it is one and  $b = 0$ , then Bob accepts  $b = 0$ . If it is zero or two and  $b = 1$ , then Bob accepts  $b = 1$ . Otherwise, he rejects  $b$ .

In spite of one-bit information, Bob obtains the  $2 \times 2$  matrix that is one of the matrices of  $\widehat{\mathcal{Z}}$ .

$$\widehat{\mathcal{Z}} = \left\{ \begin{pmatrix} 11 \\ 01 \end{pmatrix}, \begin{pmatrix} 11 \\ 10 \end{pmatrix}, \begin{pmatrix} 01 \\ 11 \end{pmatrix}, \begin{pmatrix} 10 \\ 11 \end{pmatrix}, \begin{pmatrix} 11 \\ 11 \end{pmatrix}, \begin{pmatrix} 00 \\ 11 \end{pmatrix}, \begin{pmatrix} 11 \\ 00 \end{pmatrix}, \begin{pmatrix} 10 \\ 10 \end{pmatrix}, \begin{pmatrix} 01 \\ 10 \end{pmatrix} \right\}$$

The first four matrices of  $\widehat{\mathcal{Z}}$  imply that Alice committed to  $b = 0$ , and the last five ones imply that she committed to  $b = 1$ .

We give an example. Suppose that Ted chooses  $u$  and  $v$  as follows:

$$u = \begin{pmatrix} 00 \\ 11 \end{pmatrix}, v = \begin{pmatrix} 11 \\ 10 \end{pmatrix}.$$

Ted sends  $u$  and  $v$  to Alice and Bob, respectively. We assume that Alice commits to  $b = 0$ . Alice transposes  $u$ , and permutes the rows and the columns of the transposed matrix at random. Suppose that the resulting matrix  $t$  is given by

$$t = \begin{pmatrix} 10 \\ 10 \end{pmatrix}$$

Alice sends  $t$  to Bob. In the revealing step, Alice sends  $u$  and  $b = 0$  to Bob. Bob first stacks  $u$  on  $v$ , i.e.,

$$\begin{pmatrix} 00 \\ 11 \end{pmatrix} \vee \begin{pmatrix} 11 \\ 10 \end{pmatrix} = \begin{pmatrix} 11 \\ 11 \end{pmatrix}.$$

Since all components are one, i.e., all pixels are black, Bob continues to perform the verification. Bob next stacks  $u$  on  $t$ , i.e.,

$$\begin{pmatrix} 10 \\ 10 \end{pmatrix} \vee \begin{pmatrix} 00 \\ 11 \end{pmatrix} = \begin{pmatrix} 10 \\ 11 \end{pmatrix}.$$

Since the number of white pixels is one and  $b = 0$ , Bob accepts  $b = 0$ .

It is easy to confirm the correctness and the concealment of the proposed scheme. The cheating probability of the proposed scheme is  $1/2$ . It is verified by checking all possible combinations of  $u$ ,  $v$ , and  $b$ . We can decrease the cheating probability by repeating the proposed scheme. It is effective to use the scheme described in Sect. 4.2 after transforming one-bit information into an image.

We next compare the proposed scheme to the AP scheme in terms of the communication complexity. Table 2 shows the communication complexity of both schemes. We omitted the complexity for sending  $b$  in the revealing step of both schemes. The value in the bracket of Table 2 is the essential size of data. For example, although the  $2 \times 2$  matrix that Ted sends to Alice in the initializing step seems to be 4-bit data, it is essentially 2-bit data because it must be a matrix in  $\widehat{\mathcal{U}}$ . We observe from Table 2 that the essential communication complexity of the proposed scheme is comparable to that of the AP scheme.

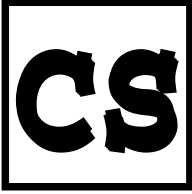


Fig. 7. Image  $\mathcal{B}$

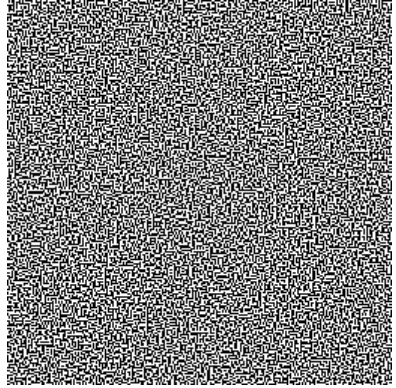


Fig. 8. Image  $\mathcal{U}$  from Ted to Alice

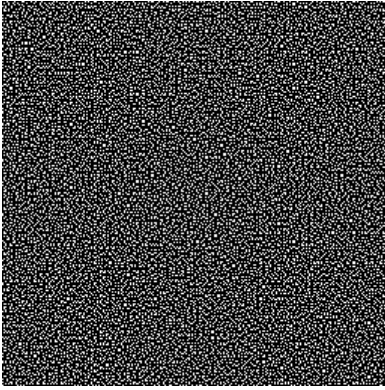


Fig. 9. Image  $\mathcal{V}$  from Ted to Bob

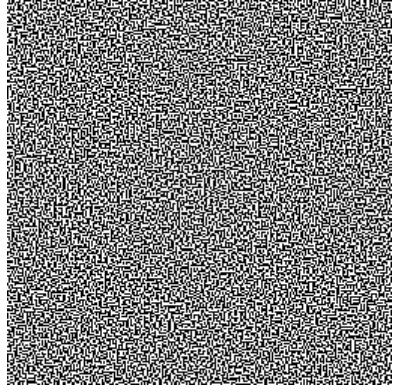


Fig. 10. Image  $\mathcal{T}$  produced by Alice



Fig. 11. Image given by  $\mathcal{U} \vee \mathcal{V}$

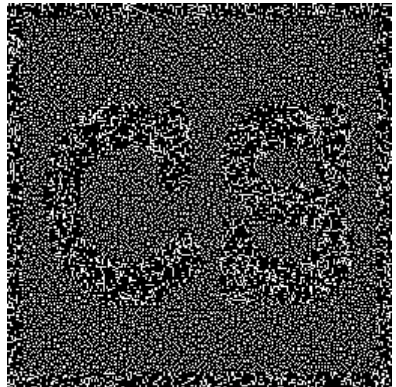


Fig. 12. Image given by  $\mathcal{U} \vee \mathcal{T}$

**Table 2.** Communication complexity in commitment schemes [bits]

Step		AP scheme	Proposed scheme
Initializing	Alice	2	4 (2)
	Bob	2	4 (2)
Committing		1	4 (2)
Revealing		2	4 (1)
Total complexity		7	16 (7)

## 4.2 Application to the Image

The visual commitment scheme of multi-bit information  $\mathcal{B}$  can be constructed by repeating that of one-bit information. Using a black and white image  $\mathcal{B}$  of Fig. 7, we give an example of the visual commitment scheme. Fig. 8 is an image  $\mathcal{U}$  of the set of  $u$ , and Fig. 9 is an image  $\mathcal{V}$  of the set of  $v$ . Fig. 10 is an image  $\mathcal{T}$  of the set of  $t$ . Fig. 11 is the image given by  $\mathcal{U} \vee \mathcal{V}$ . Since it is perfectly black, Bob visually finds that  $\mathcal{U}$  and  $\mathcal{V}$  are valid images. Fig. 12 is the image given by  $\mathcal{U} \vee \mathcal{T}$ . It follows that Bob can visually recognize the original image  $\mathcal{B}$ .

Notice that the reason that the original image can be visually recognized is different from that of the visual secret sharing scheme. In the case of the visual secret sharing scheme, the original image is visually recognized by the difference in the number of white pixels of subpixels. In the case of the proposed scheme, the original image is visually recognized by the difference in the variance of the number of white pixels of subpixels. When  $b = 0$ , the number of white pixels is always 1 and the variance is 0. When  $b = 1$  the average number of white pixels is 1 and the variance is 1. The difference in the variance enables us to recognize the original image from the recovered image.

## 5 Concluding Remarks

In this paper, we have proposed the visual oblivious transfer and the visual commitment scheme. The feature of the visual oblivious transfer is that the user can visually distinguish between success and failure. The feature of the visual commitment scheme is that the user can visually not only recognize information but also check the validity of data. To achieve these features we used the fact that the difference in the variance of the number of white pixels can be visually recognized. In addition, our results imply that the weakness of the algebraic structure causes the expansion of data transmitted in protocols.

## Acknowledgments

We would like to thank anonymous reviewers for their valuable comments.

## References

1. Blakley, G.R.: Safeguarding cryptographic keys. Proceedings of the National Computer Conference, American Federation of Information Processing Societies Proceedings **48** (1979) 313–317
2. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. Advances in Cryptology - CRYPTO '88, Lecture Notes in Computer Science **403** (1990) 27–35
3. Brickell, E.F.: Some ideal secret sharing schemes. Journal of Combinatorial Mathematics and Combinatorial Computing **6** (1989) 105–113
4. Blundo, C., Santis, A.D., Stinson, D.R., Vaccaro, U.: Graph decompositions and secret sharing schemes. Advances in Cryptology - EUROCRYPT '92, Lecture Notes in Computer Science **658** (1993) 1–24
5. Cramer, R., Fehr, S.: Optimal black-box secret sharing over arbitrary abelian groups. Advances in Cryptology - CRYPTO 2002, Lecture Notes in Computer Science **2442** (2002) 272–287
6. Shamir, A.: How to share a secret. Communications of the ACM **22** (1979) 612–613
7. Bellare, M., Micali, S.: Non-interactive oblivious transfer and application. Advances in Cryptology - CRYPTO '89, Lecture Notes in Computer Science **435** (1989) 547–559
8. Crépeau, C.: Efficient cryptographic protocols based on noisy channels. Advances in Cryptology - EUROCRYPT '97, Lecture Notes in Computer Science **1233** (1997) 306–327
9. Damgård, I., Kilian, J., Salvail, L.: On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. Advances in Cryptology - EUROCRYPT '99, Lecture Notes in Computer Science **1592** (1999) 56–73
10. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM **28** (1985) 637–647
11. Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical Report, Tech. Memo. TR-81, Aiken Computation Laboratory, Harvard University (1981)
12. Blum, M.: Coin flipping by telephone: A protocol for solving impossible problems. Advances in Cryptology - A Report on CRYPTO '81, (1982) 11–15
13. Naor, M., Shamir, A.: Visual cryptography. Advances in Cryptology - EUROCRYPT '94, Lecture Notes in Computer Science **950** (1994) 1–12
14. Rivest, R.L.: Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer.  
<http://theory.lcs.mit.edu/~rivest/Rivest-commitment.pdf> (1999)
15. Koga, H., Iwamoto, M., Yamamoto, H.: An analytic construction of the visual secret sharing scheme for color images. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E84-A** (2001) 262–272
16. Crépeau, C.: Equivalence between two flavours of oblivious transfers. Advances in Cryptology - CRYPTO '87, Lecture Notes in Computer Science **293** (1987) 350–368
17. Blundo, C., Masucci, B., Stinson, D.R., Wei, R.: Constructions and bounds for unconditionally secure non-interactive commitment schemes. Designs, Codes and Cryptography **26** (2002) 97–110

# Admissible Interference by Typing for Cryptographic Protocols

Alaaeddine Fellah and John Mullins

Département de génie informatique, École Polytechnique de Montréal,  
P.O. Box 6079, Station Centre-ville, Montréal (Québec), Canada, H3C 3A7  
{alaaeddine.fellah, john.mullins}@polymtl.ca

**Abstract.** Many security properties of cryptographic protocols can be expressed by using information flow policies as non-interference. But, in general it is very difficult to design a system without interference. For that, many works try to weak the standard definition of the non-interference. For instance, in [21] Mullins defines the admissible interference as an interference that admits flow information only through a dowgrader. Thus, we present in this paper a type system that try to detect process that allow interference. Then, if we can type a process we can say that is free interference. Also, we extend the type system of process with another type system based on a standard message algebra used in the literature of cryptographic protocols. So, we define the theoretic characterization, prove the correctness of our type system and present an illustration of our result.

**Keywords:** Admissible interference, Type systems, Process Algebra, Cryptographic Protocols, Security Properties.

## 1 Introduction

Since the advent of Internet, the list of intrusions in the information processing systems and the robbing of information via this network increase continually. Internet gave not only a perfect window to the tradesmen of the whole world who find the occasion to benefit from a virtual world market, but also caused many ideas at all those which seek easy money and those which find a great pleasure to ransack the computer sites of the others. Thus, the computer security becomes a requirement of the highest importance.

In such network, the security is enforced by the use of security protocol. The use of these protocols ensures that one or more properties of security is enforced, such as : confidentiality, authentication, integrity, non-repudiation, atomicity, etc. Nevertheless, the absence of well established formal methods to ensure the correction of the cryptographic protocols, task at the same time delicate and complex, generated undesirable consequences. Several protocols have been showed flawed many years after their publication and use.

The research on formal method related to security has increased considerably. However, based on the differences in the techniques and tools used, we can



find several disciplines in this focus of research. A complete bibliography and a comparative study of these approaches can be found in [7, 19]. In this direction, some approaches [10, 17] have used non-interference to analyze cryptographic protocols. If we consider that we have two level of security (*High, Low*), *non-interference* states that no information flow is possible from the *High*-level of security to the *Low*-level of security. So, the behavior of the users of the *High*-level (which can handle secret data) doesn't affect the behavior of the users of the *Low*-level (which handle only public data).

However, it's clear that if computation in public data not interferes with secret one, we can see that we have the perfect security. But, there is a little of system that respect this requirement. However, in the least year many works [1, 2, 11, 21] have tried to weak the standard non-interference, by putting some restriction in order to adapt this information flow policies more practice in some cases. The common idea is to ensure that program leaks sensitive information only through many conditions relative to the design of the system.

We present in this paper a type system that try to detect process that not respect the admissible interference. Our type system is composed of two parts : The first part is more general and tries to type a process. In this part we consider only standard interference, and we suppose that we have the type of the actions that constitute the process. We proof the correctness of our type system. Indeed, we proof that if a process can be typed, we can conclude that this process is free interference. This result is general and not depends on the structure of any system. The second part shows how to type exchanged message in a cryptographic protocol, and then how to type actions that may constitute a process. Thus, we consider implicitly admissible interference in the typing of encrypted message.

The remainder of the paper is organized as follows: In the section 2 we present the process algebra used in rest of this paper. In the section 3 we present the type system of process, and we proof the correctness of it. In the section 4 we extend the type system of process with another type system based on a standard message algebra used in the literature of cryptographic protocols. In the section 5 we present an example to illustrate our approach. In the section 6 we present several related works. We end by a conclusion and possible amelioration of this work.

## 2 Process Algebra

### 2.1 Syntax

The set of all process, denoted by  $\mathcal{P}$ , is defined by the following grammar :

$P ::=$	<i>Stop</i>	<b>Stop</b>	$a.P$	<b>Prefix</b>
	$P \setminus N$	<b>Restriction</b>	$P[F]$	<b>Relabelling</b>
	$P/L$	<b>Masquing</b>	$Z$	<b>Identifier</b>
	$P  P$	<b>Parallel</b>	$P + P$	<b>Choice</b>

The principal entity of a process is the action. We suppose that we have a set of action, denoted by  $\mathcal{Act}$ , and contain output action  $(a, b, \dots)$  and input action  $(\bar{a}, \bar{b}, \dots)$ , with  $(\bar{\bar{a}} = a)$ . Latter in this paper, we give more detail in the syntax of the action used specially in cryptographic protocols.

$Stop$  is the process which doesn't do anything.  $\alpha.P$  is the process which is ready to execute the action  $\alpha$  and then to behave like  $P$ .  $P \setminus N$  is the process which behaves as  $P$ , but it can't execute any action of the set  $L$ .  $P/L$  is the process which behaves as  $P$ , but it can execute only the action of the set  $L$ .  $P[F]$  is the process resulting from the application of the relabelling function  $F$  on the actions which constitute the process  $P$ .  $Z$  is a constant that must be associated to a definition of a process.  $P \parallel Q$  is the parallel composition of the two process  $P$  and  $Q$ .  $P + Q$  is the choice composition of the process  $P$  and  $Q$ .

## 2.2 Operational Semantics

The operational semantics is defined through the evolution relation  $\longrightarrow$  which we will define on the set of processes. However, we need to arrange the processes in a form which allows us to use the evolution relation. For that, we will define the structural equivalence, noted  $\equiv$ , as the smallest relation which satisfies the rules given in the table 1.

**Table 1.** Structural equivalence

$P \parallel 0 \equiv P$	$0[F] \equiv 0$	$P + Q \equiv Q + P$	$(P + P') \setminus H \equiv P \setminus H + P' \setminus H$
$P + 0 \equiv P$	$0 \setminus N \equiv 0$	$P \parallel Q \equiv Q \parallel P$	

The structural equivalence show that when the process 0 is used in a choice, parallel communication, we can simplify the whole process by eliminating the process 0. Also, when we applied the relabelling or restriction operators on the process 0 the whole process is equivalent to the process 0. We note that the choice and parallel operators are commutative.

The operational semantics is given in the table 2. The label used in this semantic belong to  $\mathcal{Act} \cup \{\tau\}$ , with  $\tau$  represents an inter evolution. Then, in this table  $a \in \mathcal{Act}$ ,  $\bar{b}$  an input action,  $b'$  an output action,  $N$  and  $L$  sets of actions and  $\sigma$  a substitution.

## 3 Typing System for Non-interference

### 3.1 Types

We presuppose a complete lattice of level of security  $(\Sigma, \leq)$  ranged over  $\alpha, \beta, \dots$ . The order relation  $\leq$  determines an order in the level of security. Thus,

**Table 2.** Operational semantics

$R$	$\frac{}{a.P \xrightarrow{a} P}$	$R_Z$	$\frac{P \xrightarrow{a} P'}{Z \xrightarrow{a} P'} [Z \stackrel{def}{=} P]$
$R_{\setminus}$	$\frac{P \xrightarrow{a} P'}{P \setminus N \xrightarrow{a} P' \setminus N} [a \notin N]$	$R_f$	$\frac{P \xrightarrow{a} P'}{P[F] \xrightarrow{F(a)} P'[F]}$
$R_{/1}$	$\frac{P \xrightarrow{a} P'}{P/L \xrightarrow{a} P'/L} [a \notin N]$	$R_{/2}$	$\frac{P \xrightarrow{a} P'}{P/L \xrightarrow{\tau} P'/L} [a \in N]$
$R_{\parallel}^s$	$\frac{P \xrightarrow{\bar{b}} P' \quad Q \xrightarrow{b'} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} [\exists \sigma. b = b'\sigma]$	$R_{\parallel}^e$	$\frac{P \xrightarrow{a} P'}{P \parallel Q \xrightarrow{a} P' \parallel Q}$
$R_+$	$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'}$	$R_{\equiv}$	$\frac{P \equiv P' \quad P' \xrightarrow{a} Q' \quad Q' \equiv Q}{P \xrightarrow{a} Q}$

an element  $\tau$  is less than the element  $\tau'$ , if the first represents a level of security less than the second. In other terms, an element of type  $\tau'$  is more secret than an element of type  $\tau$ . Also, the lattice  $\Sigma$  has  $\top$  as the greatest element (the high level of security) and  $\perp$  as the lowest element (the low level of security).

We denote also the type of an action by  $\alpha pro$  with  $\alpha \in \Sigma$  and  $pro$  a suffix to distinguish between type of action and type of message (that we will present later in this article). The goal of the type system is to type process and to detect if this process contains interference. In the type system, the behavior of a process is related to a static environment. This environment, denoted by  $\gamma$ , associates a value to a type. The values of the domain of the environment represent the knowledge necessary and sufficient to be able to type a process. For example, to type a process which represents a cryptographic protocol, we will need to know the type of all the atomic messages used in this protocol. Then, we use judgements of the form :

$$\Gamma \vdash P : \alpha$$

with  $P \in \mathcal{P}$  and  $\alpha \in \Sigma_P$ . This judgement means that relative to the environment  $\Gamma$ , the processus  $P$  has the type  $\alpha$ . This type correspond to the greatest type of the actions which constitute  $P$ . Also, we can say that the process  $P$  is free interference. We can classify the process according to there types, then we denote the set  $\mathcal{P}_\alpha$  the set of process whom have the type  $\alpha pro$ .

In this paper we investigate two relations :  $\alpha$ -bisimulation and  $(\alpha, \beta)$ -test equivalence. The  $\alpha$ -bisimulation is an extension of the standard bisimulation. In this new relation the bisimulation is made according to a type of actions. Two process are  $\alpha$ -bisimilar if there behaviors are equivalents in the level  $\alpha$ . More formally :

**Definition 1 ( $\alpha$ -bisimulation).**

Let  $P, Q \in \mathcal{P}$ ,  $\alpha \in \Sigma$ . we say that  $P \sqsubseteq^\alpha Q$  if :

1.  $\forall a \in \text{Act}_\alpha$  if  $P \xrightarrow{a} P'$  then  $\exists \delta \in \overline{\text{Act}_\alpha}^*$ .  $Q \xrightarrow{\delta, a} Q'$  with  $P' \sqsubseteq^\alpha Q'$
2.  $\forall \delta \in \overline{\text{Act}_\alpha}^*$  if  $P \xrightarrow{\delta} P'$  then  $\exists \delta' \in \overline{\text{Act}_\alpha}^*$ .  $Q \xrightarrow{\delta'} Q'$  with  $P' \sqsubseteq^\alpha Q'$

Thus,  $P$  and  $Q$  are  $\alpha$ -bisimilar, and we denote  $P \approx^\alpha Q$  if :

$$P \sqsubseteq^\alpha Q \wedge Q \sqsubseteq^\alpha P$$

From this relation, we can define an other relation that extend the test equivalence :  $(\alpha, \beta)$ -test equivalence. In this relation, the test is a process that has the type  $\beta$ . Thus, two process are  $(\alpha, \beta)$ -test equivalence if for all test of type  $\beta$ , the parallel composition of the two process with the test are  $\alpha$ -bisimilar. More formally :

**Definition 2 ( $(\alpha, \beta)$ -test equivalence).**

Let  $P, Q \in \mathcal{P}$ ,  $\alpha, \beta \in \Sigma$ . we say that  $P \sqsubseteq_\beta^\alpha Q$  if :

$$\forall R \in \mathcal{P}_\beta : P || R \sqsubseteq^\alpha Q || R$$

Thus,  $P$  and  $Q$  are  $(\alpha, \beta)$ -test equivalent, and we denote  $P \approx_\beta^\alpha Q$  if :

$$P \sqsubseteq_\beta^\alpha Q \wedge Q \sqsubseteq_\beta^\alpha P$$

The definition which follows define another relation that extend the  $(\alpha, \beta)$ -test equivalence to associate actions and process. We say that the action  $a$  don't cause the process  $P$ , if this action don't influence the behavior of the process  $P$ . More formally:

**Definition 3 (Causality).** Let  $P \in \mathcal{P}$ ,  $a \in \text{Act}$ ,  $\alpha \in \Sigma$ . we note  $a \overset{\alpha}{\rightsquigarrow} P$  if :

$$P \sqsubseteq_\alpha^\alpha a.P$$

**3.2 Typing System**

The rules of the type system are given in the table 3. We note that we doesn't need to define rule for each operator, some rule are deductible from the two rules  $R_{.1}, R_{.2}, R_{\equiv}$ . The intuitive idea underlying each rule are given as follow :

$R_{.1}$  **et**  $R_{.2}$ : These two rules allow us to type prefixed processes. Thus, if we suppose that we have an action  $a$  of type  $\alpha pro$  and a process  $P$  of the type  $\alpha' pro$ . Then, the rule  $R_{.1}$  ensures that if the level of security  $\alpha$  is lower than the level  $\alpha'$  then the process  $a.P$  will have the type  $\alpha pro$ . Also, the condition of this rule ( $\alpha \preceq \alpha'$ ) ensures us that there isn't implicit flow of information. On the other hand, the rule  $R_{.2}$  affirms that if the level  $\alpha$  is higher than the level  $\alpha'$ , then we must make sure that there is not a dependence between the action  $a$  and the process  $P$ . The condition  $a \overset{\alpha}{\rightsquigarrow} P$  ensures that. Thus, we can say that the process  $a.P$  has the type  $\alpha' pro$  and it is free interference.

**Table 3.** Typing System

$(R_{\cdot 1})$	$\frac{\Gamma \vdash a : \alpha pro \quad \Gamma \vdash P : \alpha' pro}{\Gamma \vdash a.P : \alpha pro} [\alpha \preceq \alpha']$		
$(R_{\cdot 2})$	$\frac{\Gamma \vdash a : \alpha pro \quad \Gamma \vdash P : \alpha' pro}{\Gamma \vdash a.P : \alpha' pro} [\alpha' \preceq \alpha \wedge a \overset{\alpha}{\rightsquigarrow} P]$		
$(R_{  })$	$\frac{\Gamma \vdash a : \alpha pro \quad \Gamma \vdash a' : \alpha' var \quad \Gamma \vdash P    Q \sigma : \beta pro}{\Gamma \vdash a.P    a'.Q : \beta pro} [\alpha \preceq \alpha' \wedge a = \overline{a'} \sigma]$		
$(R_{[\_ ]})$	$\frac{\Gamma \vdash P : \alpha pro}{\Gamma \vdash P[F] : \alpha pro}$	$(R_{+})$	$\frac{\Gamma \vdash P : \alpha pro \quad \Gamma \vdash P' : \alpha pro}{\Gamma \vdash P + P' : \alpha pro}$
$(R_{\mathbf{0}})$	$\frac{}{\Gamma \vdash \mathbf{0} : \top pro}$	$(R_{\equiv})$	$\frac{\Gamma \vdash P : \alpha pro \quad P \equiv Q}{\Gamma \vdash Q : \alpha pro}$

$R_{||}$ : This rule allows us to type processes with synchronization. Thus, if a sent action  $a$  has the type  $\alpha pro$ , the received corresponding action  $a'$  has the type  $\alpha' pro$  and if the level  $\alpha$  is lower than the level  $\alpha'$ , then the process  $a.P || a'.Q$  has the same type as the process  $P || Q \sigma$ , with  $\sigma$  is the substitution which equalizes the action  $a$  with the action  $a'$ . With the condition  $\alpha \preceq \alpha'$  we ensure that we don't have interference in the process.

$R_{[\_ ]}$ : If the process  $P$  is of type  $\alpha pro$  then the process  $P[F]$  has the same type. The type is safeguarded since the function of renaming is ensured to re-elect terms in the same way standard.

$R_{\mathbf{0}}$ : The process  $\mathbf{0}$  has the  $\top$  type. This is only a choice to avoid having interferences between the high level process  $\mathbf{0}$  and any action.

$R_{\equiv}$ : This rule affirms that if two processes are equivalent structurally then they have the same type.

### 3.3 Interference by Typing

In this section, we use the type system given in the preceding section to give a result on the interference. The definition that we will use for the interference is based on the basic idea of the non-interference given in [12]: "No flow of information is possible of a high level user to another of low level". We denote the set of all actions that have the type  $\top pro$  by  $\mathcal{H}$ . Thus, we characterize formally the non-interference as follows:

#### Definition 4 (Non-interference).

Let  $P$  a process. We say that  $P$  is free interference, and we note  $\mathcal{IF}(P)$ , if :

$$(P/\mathcal{H}) \sqsubseteq_{\perp}^{\perp} P$$

This definition say that all the behaviors of the process  $P/\mathcal{H}$  in the low level (action with the type  $\perp$ ) are a behavior of the process  $P$ . Then, the behavior of the high level doesn't influence the one in the low level.

**Theorem 1 (Correctness).** *Let  $P$  a process and  $\Gamma$  a static environment.*

$$\text{If } \exists \alpha \in \Sigma . \Gamma \vdash P : \alpha \text{ pro then } \mathcal{IF}(P)$$

*Proof.* The proof of the theorem and some propositions used to proof it is available in the web page of the authors in the full version of this paper (<http://www.polymtl.ca/crac/>).

## 4 Admissible Interference in Cryptographic Protocols

In this section we will extend our process algebra with an algebra of message specific to the cryptographic protocols. Indeed, the goal is to check the non interference in the cryptographic protocols and to extend this verification to verify some properties of security such as : confidentiality and authentication.

We use the standard message algebra used in the specification of cryptographic protocols literature. The set of all messages, denoted by  $\mathcal{M}$ , is defined by the grammar given in the table 4. So, a message can be the identity of an agent, a nonce, a key, a variable. We use also two operators: the pair and the encryption.

**Table 4.** Messages

$M ::= A$	Agent	$N$	Nonce
$  K$	Key	$x$	Variable
$  (M_1, M_2)$	Pair	$\{M\}_K$	Encryption

After giving the syntax of the messages, we can detail the form of the actions. We consider only two type of action : the sent action and the received action. Thus, an action is identified by three things: The name of the channel of communication, the type of the action and the message exchanged. Thus, the set of all actions  $\mathcal{Act}$  is defined as follows :

$$\underline{a ::= \nu_{xy}^\alpha(m) \quad \text{Send action} \quad | \quad \overline{\nu_{xy}^\alpha(m)} \quad \text{Reception action}}$$

The action  $\nu_{xy}^\alpha(m)$  is the action that send the message  $m$  in the channel charred between the agent  $x$  and  $y$  during the session  $\alpha$ . The same for the reception action  $\overline{\nu_{xy}^\alpha(m)}$ .

The type  $\alpha$  of a message represents the level of security of it ( $\alpha \in \Sigma$ ), and it's controlled by two facts : The kind of the action which will handle this message (reception or send action) and the agent which makes the action. Indeed, the same message can have two different type dependently of the kind of the action. Then, we use judgment of the form :

$$\Gamma \vdash_s^a M : \alpha$$

we read this : In the static environment  $\Gamma$  the message  $m$  sent by the agent  $A$  has the type  $\alpha$  ( $\alpha \in \Sigma$ ). And also :

$$\Gamma \vdash_r^a M : \alpha$$

which we read : In the static environment  $\Gamma$  the message received by the agent  $A$  has the type  $\alpha$  ( $\alpha \in \Sigma$ ). When the kind of the action is not specified, this means that the message to be typed has the same type if it is sent or received. In the table 6 we present the rules to type messages.

**Table 5.** Typing rules for messages

$(RT_{ato}) \quad \frac{\mathcal{E}(M) = \alpha}{\Gamma \vdash_r^a M : \alpha} [M \in \mathcal{M}_{ato}]$	$(RT_{pair}) \quad \frac{\Gamma \vdash^a M_1 : \alpha_1 \quad \Gamma \vdash^a M_2 : \alpha_2}{\Gamma \vdash^a (M_1, M_2) : sup(\alpha_1, \alpha_2)}$
$(RT_{enc1}) \quad \frac{\Gamma \vdash_s^a M : \alpha \quad \Gamma \vdash_s^a k^{-1} : \top}{\Gamma \vdash_s^a \{M\}_k : inf(\mathcal{D}, \alpha)} [k \in C_a]$	$(RT_{enc2}) \quad \frac{\Gamma \vdash_s^a M : \alpha \quad \Gamma \vdash_s^a k^{-1} : \perp}{\Gamma \vdash_s^a \{M\}_k : \alpha} [k \in C_a]$
$(RT_{enc3}) \quad \frac{}{\Gamma \vdash_s^a \{M\}_k : \perp} [k \notin C_a]$	$(RT_{sig}) \quad \frac{\Gamma \vdash_s^a M : \alpha}{\Gamma \vdash_s^a \{M\}_{k^{-1}} : \alpha}$
$(RT_{dec1}) \quad \frac{\Gamma \vdash_r^a M : \alpha \quad \Gamma \vdash_r^a k : \top}{\Gamma \vdash_r^a \{M\}_k : \alpha} [k \in C_a]$	$(RT_{dec2}) \quad \frac{\Gamma \vdash_r^a k^{-1} : \perp}{\Gamma \vdash_r^a \{M\}_k : \perp} [k \in C_a]$
$(RT_{dec3}) \quad \frac{}{\Gamma \vdash_r^a \{M\}_k : \perp} [k \notin C_a]$	

The type of the messages differs between the  $\perp$  and the  $\top$  type. It is the case also for the key of encryption and decryption. Thus, in order to treat the cryptographic protocol with symmetrical or asymmetrical key, we give a particular signification to the type of the keys. If the type of a key is  $\top$ , that means that this key is a secret key which is not known by the intruder. However, if the key has the  $\perp$  type, this means that either it is a public key or it is a key used by an intruder. Thus, we distinguish between the last two cases by using the key of decryption. Then, if the key of decryption, corresponding to a key of encryption of the  $\perp$  type, has the  $\top$  type, we conclude that it's a honest agent key. In the contrary, we conclude that the key is used by an intruder. The intuitive signification of each rule of the table 6 is given as follow :

$RT_{ato}$ : The type of an atomic message (identities, key and nonce) is the type associates to it in the static environment.

$RT_{pair}$ : The type of a pair is the upper limit of the type of its components.

$RT_{enc1}$ : If the type of a message  $M$  sent by an agent  $A$  is  $\alpha$ , and the key  $k$  has the  $\top$  type and belongs to the knowledge of  $A$ , then the message  $M$  encrypted by the key  $k$  sent by the agent  $A$  has the type  $\perp$ . Thus, if the message  $M$  has the type  $\top$  then when we encrypt it with a key which doesn't belong to an intruder, we downgrade these data and associate them the type  $\perp$ . Consequently, we admit implicitly the interference which can be between the secret message  $M$  (type  $\top$ ), and the encrypted message (type  $\perp$ ).

$RT_{enc2}$ : If a message is encrypted with the key of the intruder, then we consider that the message is sent without encryption. Thus, the encrypted message with the key of the intruder has the same type that the message not encrypted.

$RT_{enc3}$ : If an agent sends an encrypted message without having the key of encryption, this means that he received it before, and that he only replicate it. Thus, we associate the low level type to this message.

$RT_{dec1}$ : If a message  $M$  has the type  $\alpha$  and it is encrypted by a high level key  $k$  which belongs to agent  $A$ , then the message  $\{m\}_k$  received by the agent  $A$  has the type  $\alpha$  because the agent  $A$  has the capacity of decrypting it.

$RT_{dec2}$ : If an agent receives an encrypted message which he doesn't have the key of decryption then this message has the type  $\perp$ .

$RT_{dec3}$ : If an agent receives an encrypted message which the key of encryption has the type  $\perp$  and it belong to the knowledge of this agent, then this message has the type  $\perp$ .

Also we must extend the type system given in the table 3 by the two following rule, to be able to type action from the type of the message exchanged in this action.

**Table 6.** Typing rules for actions

$(RT_{send}) \quad \frac{\Gamma \vdash_s^\alpha M : \alpha}{\Gamma \vdash \nu_{ax}(M) : \alpha pro}$	$(RT_{recp}) \quad \frac{\Gamma \vdash_r^\alpha M : \alpha}{\Gamma \vdash \bar{\nu}_{xa}(M) : \alpha pro}$
--	--

## 5 Example: The Woo and Lam Public Key Protocol

### 5.1 Specification of the Protocol

The Woo-Lam public key protocol is defined in the table 7. We have three rules in this protocol : the rule of the agent  $A$  (Initiator), the rule of the agent  $B$  (Responder) and the rule of the server. We model the processes of the rule of the agent  $A$  is specified as follows :

**Table 7.** Woo-Lam public key protocol

- 1  $A \rightarrow S : A, B$
- 2  $S \rightarrow A : \{K_B\}_{K^{-1}}$
- 3  $A \rightarrow B : \{A, N_A\}_K$
- 4  $B \rightarrow S : A, B, \{N_A\}_K$
- 5  $S \rightarrow B : \{K_A\}_{K^{-1}}, \{\{N_A, K, A, B\}_{K^{-1}}\}_K$
- 6  $B \rightarrow A : \{\{N_A, K, A, B\}_{K^{-1}}, N_B\}_K$
- 7  $A \rightarrow B : \{N_B\}_K$



$$Init(A) := C_{ar}^\alpha(A, B). \overline{C}_{ra}^\alpha(\{X\}_{k^{-1}}). C_{ar}^\alpha(\{A, N_A\}_X). \\ \overline{C}_{ra}^\alpha(\{\{N_A, Y, A, B\}_{K^{-1}}, Z\}_K). C_{ar}^\alpha(\{Z\}_Y)$$

The other roles (Responder, server) are specified in a similar way.

We need to model a malicious intruder that participates to the protocol. We suppose that the intruder has the all control in the network : he can intercept messages, create and send any message from his base of knowledge. However, for simplify the example we use the following process to represent the intruder :

$$I := \overline{C}_{rb}^\alpha(\{Y\}_{K^{-1}}, \{\{X, Z, A, B\}_{K^{-1}}\}_K). C_{sr}^\alpha(\{K_I\}_{K^{-1}}, \{\{X, Z, A, B\}_{K^{-1}}\}_K). \\ \overline{C}_{ra}^\alpha(\{\{X, Z, A, B\}_{K^{-1}}, Z\}_K). C_{br}^\alpha(\{\{X, Z, A, B\}_{K^{-1}}, N_B\}_Y)$$

Thus the process that represents the protocol is the parallel composition of all the agents that participate at it.

## 5.2 Verification of the Protocol

The goal of the verification is to detect if the process  $Pro$  is free interference. For that we try to find a type  $\alpha$  such as  $\Gamma \vdash P : \alpha_{pro}$ . For this example, it is sufficient to use only one copy of each agent ( $Init(A_1), Resp(B_1)$ ). For simplify the proof tree we denote  $Init(A_1)$ ,  $Resp(B_1)$  and  $Ser$  by  $A_1$ ,  $B_1$  and  $S$  respectively. We use the following static environment:

$$\Gamma = \{A \mapsto \perp, B \mapsto \perp, S \mapsto \perp, I \mapsto \perp, K_A \mapsto \perp, K_B \mapsto \perp, K_S \mapsto \perp, K_I \mapsto \perp \\ K_A^{-1} \mapsto \top, K_B^{-1} \mapsto \top, K_S^{-1} \mapsto \top, K_I^{-1} \mapsto \perp, N_A \mapsto \perp, N_B \mapsto \perp, K \mapsto \top\}$$

The table 8 gives the type of some messages exchanged in the protocol.

Thus the proof tree for the typing of the process  $Pro$  is given in the table 9. In this tree, each time that a process evolves, we will change its indexation. Also, to not encumber the proof tree, we will not put the conditions of the rules if they are checked (such as for the rule  $R_{||}$ ).

However, we will not be able to type this process. Indeed, in the tree  $Arb_7$  we find that the message sent  $\{\{n_a, K, has, b\}_{k^{-1}}, N_B\}_K$  by the agent  $B$  has the type  $\top$ , and the message received by the intruder  $\{\{n_a, K, has, b\}_{k^{-1}}, N_B\}_K$  has the type  $\perp$ . Since the condition of the rule  $R_{||}$  imposes that the type of the send action must be lower than the type of the reception action. Thus, we can say that the process  $Pro$  is not free interference. This problem led to a known flaw of confidentiality in this protocol [8].

**Table 8.** Type of some messages exchanged

$\Gamma \vdash_s^i \{K_I\}_{K_S^{-1}}, \{\{N_A, K, A, B\}_{K_S^{-1}}\}_{K_B} : \perp$	$\Gamma \vdash_r^s \{K_I\}_{K_S^{-1}}, \{\{N_A, K, A, B\}_{K_S^{-1}}\}_{K_B} : \perp$
$\Gamma \vdash_s^b \{\{N_A, K, A, B\}_{K_S^{-1}}, N_B\}_{K_I} : \top$	$\Gamma \vdash_r^i \{\{N_A, K, A, B\}_{K_S^{-1}}, N_B\}_{K_I} : \perp$

Table 9. Proof tree

$\frac{\Gamma \vdash_s^a A, B : \perp}{\Gamma \vdash C_{sa}^a(A, B) : \perp_{pro}} \quad R_{act} \quad \frac{\Gamma \vdash_r^a A, B : \perp}{\Gamma \vdash \overline{C}_{sa}^a(A, B) : \perp_{pro}} \quad R_{act} \quad \frac{}{\Gamma \vdash A_1    B_1    S_1    I_1 : ?} \quad R_{  }}{\Gamma \vdash A_1    B_1    S_1    I_1 : ?} \quad R_{  } \quad Arb_1$
$\frac{\Gamma \vdash_s^a \{K_B\}_{K_S^{-1}} : \perp}{\Gamma \vdash C_{sa}^a(\{K_B\}_{K_S^{-1}}) : \perp_{pro}} \quad R_{act} \quad \frac{\Gamma \vdash_r^a \{K_B\}_{K_S^{-1}} : \perp}{\Gamma \vdash \overline{C}_{sa}^a(\{K_B\}_{K_S^{-1}}) : \perp_{pro}} \quad R_{act} \quad \frac{}{\Gamma \vdash A_2    B_1    S_2    I_1 : ?} \quad R_{  }}{\Gamma \vdash A_2    B_1    S_2    I_1 : ?} \quad R_{  } \quad Arb_2$
$\frac{\Gamma \vdash_s^a \{A, N_A\}_{K_B} : \perp}{\Gamma \vdash C_{sa}^a(\{A, N_A\}_{K_B}) : \perp_{pro}} \quad R_{act} \quad \frac{\Gamma \vdash_r^b \{A, N_A\}_{K_B} : \perp}{\Gamma \vdash \overline{C}_{sa}^a(\{A, N_A\}_{K_B}) : \perp_{pro}} \quad R_{act} \quad \frac{}{\Gamma \vdash A_3    B_1    S_3    I_1 : ?} \quad R_{  }}{\Gamma \vdash A_3    B_1    S_3    I_1 : ?} \quad R_{  } \quad Arb_3$
$\frac{\Gamma \vdash_s^b A, B, \{N_A\}_{K_S} : \perp}{\Gamma \vdash C_{sb}^a(A, B, \{N_A\}_{K_S}) : \perp_{pro}} \quad R_{act} \quad \frac{\Gamma \vdash_r^a A, B, \{N_A\}_{K_S} : \perp}{\Gamma \vdash \overline{C}_{sb}^a(A, B, \{N_A\}_{K_S}) : \perp_{pro}} \quad R_{act} \quad \frac{}{\Gamma \vdash A_4    B_2    S_3    I_1 : ?} \quad R_{  }}{\Gamma \vdash A_4    B_2    S_3    I_1 : ?} \quad R_{  } \quad Arb_4$
$\frac{\Gamma \vdash_s^b \{K_A\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B : \perp}{\Gamma \vdash C_{sa}^a(\{K_A\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B) : \perp_{pro}} \quad R_{act} \quad \frac{\Gamma \vdash_r^i \{K_A\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B : \perp}{\Gamma \vdash \overline{C}_{sa}^a(\{K_A\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B) : \perp_{pro}} \quad R_{act} \quad \frac{}{\Gamma \vdash A_4    B_3    S_4    I_1 : ?} \quad R_{  }}{\Gamma \vdash A_4    B_3    S_4    I_1 : ?} \quad R_{  } \quad Arb_5$
$\frac{\Gamma \vdash_s^i \{K_I\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B : \perp}{\Gamma \vdash C_{sb}^a(\{K_I\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B) : \perp_{pro}} \quad R_{act} \quad \frac{\Gamma \vdash_r^b \{K_A\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B : \perp}{\Gamma \vdash \overline{C}_{sb}^a(\{K_A\}_{K_S^{-1}}, \{(N_A, K, A, B)\}_{K_S^{-1}} K_B) : \perp_{pro}} \quad R_{act} \quad \frac{}{\Gamma \vdash A_4    B_3    S_5    I_2 : ?} \quad R_{  }}{\Gamma \vdash A_4    B_3    S_5    I_2 : ?} \quad R_{  } \quad Arb_6$
$\frac{\Gamma \vdash_s^b \{(N_A, K, A, B)\}_{K_S^{-1}}, N_B \}_{K_I} : \top}{\Gamma \vdash C_{sa}^a(\{(N_A, K, A, B)\}_{K_S^{-1}}, N_B \}_{K_I}) : \top_{pro}} \quad R_{act} \quad \frac{\Gamma \vdash_r^i \{(N_A, K, A, B)\}_{K_S^{-1}}, N_B \}_{K_I} : \perp}{\Gamma \vdash \overline{C}_{sa}^a(\{(N_A, K, A, B)\}_{K_S^{-1}}, N_B \}_{K_I}) : \perp_{pro}} \quad R_{act} \quad \frac{}{\Gamma \vdash A_4    B_4    S_5    I_3 : ?} \quad R_{  }}{\Gamma \vdash A_4    B_4    S_5    I_3 : ?} \quad R_{  } \quad Arb_7$
$[Cond]R_{  }$

## 6 Related Work

The research on formal method related to security has increased considerably. However, based on the differences in the techniques and tools used, we can find several disciplines in this focus of research. A complete bibliography and a comparative study of these approaches can be found in [19, 5, 20, 7].

Recently, a promising new approach has been developed for the analyze of the information flow : The use of the type systems. A general survey of these approaches can be found in [22]. In a security typed language, the standard type of expression is increased by some annotations that specify that now information of a high level can influence the observation of low level. In this direction we found several work [9, 13, 16].

For instance, in [13] the authors are extended the asynchronous  $\pi$ -calculus [6, 15] with types: The security  $\pi$ -calculus. The authors use a mild variation of the  $I$ -types of [14]; essentially types are sets of read/write capabilities, where in addition each capability is annotated by a security level. So, the behavior of a process is relative to a security level. Then, Hennessy enforce the notion of non-interference by using a *must* test equivalence.

Type system are also used for the verification of other properties such that confidentiality and authentication. For instance we found [4, 18]. In [4], Abadi extends the spi-calculus [3] with type system that guarantee confidentiality. In

this approach messages and channels are assigned different types, and security flaws are identified as type violations. In [18], the authors use a type system that are based really in a inference system. Indeed, every message is associated to a sequence of messages that an intruder can send to have it. Thus, the types used is very sophisticated, compared to general type, and are based on a inference system that infers the knowledge of an intruder.

Also, many approaches [10, 17] have used non-interference to analyze cryptographic protocols. In [10] a wide range of security properties have been shown to be expressible in terms of non-interference. In [17] the authors express security properties by using the admissible interference [21], that is expressed by simply identifying downgrading actions corresponding to encryption actions occurring in a protocol.

## 7 Conclusion and Future Work

In this paper we present a correct type system to detect non interference in a process. Also, we extend this type system to verify admissible interference in cryptographic protocol. The rules for typing message in cryptographic are oriented to the verification of the secrecy property. However, we will extend this work by giving an uniform framework to the verification of some properties of security as confidentiality, authentication and denial of service. This framework will be based on a general definition of this properties from the admissible interference. Consequently, we will be able to verify all this properties with the same type system. Also, we intend to give a sound inference algorithm that mechanizes the type system given in this paper.

## References

1. C. Piazza A. Bossi and S. Rossi. Modelling downgrading in information flow security. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, page 187. IEEE Computer Society, 2004.
2. A. Sabelfeld A. C. Myers and S. Zdancewic. Enforcing robust declassification. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop*, pages 172–186, June 2004.
3. M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*. ACM Press, April 1997.
4. Martin Abadi. Secrecy by typing in security protocols. *J. ACM*, 46(5):749–786, 1999.
5. G. Bella. *Inductive Verification of Cryptographic Protocols*. PhD thesis, University of Cambridge, Mars 2000.
6. G. Boudol. Asynchrony and the  $\pi$ -calulus. Technical report, INRIA-Sophia Antipolis, 1992.
7. L. Buttyán. Formal methods in the design of cryptographic protocols. Technical Report SSC/1999/038, Institute for computer Communications and Applications, November 1999.

8. U. Carlsen. Cryptographic Protocol Flaws. In *Proceedings of the IEEE Computer Security Foundations Workshop VII, Franconia*, pages 192–200. IEEE, June 1994.
9. S. Conchon. Modular information flow analysis for process calculi. In *Proc. of Foundations of Computer Security*, pages 23–34, 2002.
10. R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In *Foundations of Security Analysis and Design - Tutorial Lectures. LNCS 2171.*, pages 331–396. Springer, 2001.
11. Roberto Giacobazzi and Isabella Mastroeni. Abstract non-interference: parameterizing non-interference by abstract interpretation. In *POPL '04: Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 186–197. ACM Press, 2004.
12. J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proceedings of the 1982 IEEE Symposium on Research in Security and Privacy*, pages 11–20. IEEE press, April 1982.
13. M. Hennessy. The Security Picalculus and Non-interference. *Journal of Logic and Algebraic Programming*, 2003. To Appear.
14. M. Hennessy and J. Riely. Information flow vs. resource access in the asynchronous  $\pi$ -calculus. *ACM Transactions on Programming Languages and Systems*, 24(5):566–591, September 2002.
15. K. Honda and M. Tokoro. On asynchronous communication semantics. In *Proceedings of the ECOOP'91 Workshop on Object-Based Concurrent Computing*, volume 612, pages 21–51. Springer-Verlag, 1992.
16. K. Honda and N. Yoshida. A Uniform Type Structure for Secure Information Flow. *ACM SIGPLAN Notices*, 37(1):81–92, January 2002.
17. S. Lafrance and J. Mullins. Bisimulation-based non-deterministic admissible interference and its application to the analysis of cryptographic protocols. In James Harland, editor, *Electronic Notes in Theoretical Computer Science*, volume 61. Elsevier Science Publishers, 2002.
18. M. Mejri M. Debbabi, N. A. Durgin and J. C. Mitchell. Security by typing. *International Journal on Software Tools for Technology Transfer*, 4(4):472–495, 2003.
19. C. Meadows. Formal methods for cryptographic protocol analysis: emerging issues and trends, 2003.
20. M. Mejri. *From type theory to the verification of security protocols*. PhD thesis, Laval University, February 2001.
21. J. Mullins. Nondeterministic Admissible Interference. *Journal of Universal Computer Science*, 6(11):1054–1070, November 2000.
22. A. Sabelfeld and A. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communications*, 21(1), 2003.

# On the Security Bounds of CMC, EME, EME<sup>+</sup> and EME\* Modes of Operation

Raphael C.-W. Phan<sup>1</sup> and Bok-Min Goi<sup>2,\*</sup>

<sup>1</sup> Information Security Research (iSECURES) Lab,  
Swinburne University of Technology (Sarawak Campus),  
93576 Kuching, Malaysia  
rphan@swinburne.edu.my

<sup>2</sup> Centre for Cryptography and Information Security (CCIS),  
Faculty of Engineering, Multimedia University,  
63100 Cyberjaya, Malaysia  
bmgoi@mmu.edu.my

**Abstract.** Since 2002, variants of two tweakable block cipher modes of operation, CMC and EME, have been presented by Halevi and Rogaway that are suitable for encryption of disk sectors. In this paper, we show that the security bounds given in their proofs are tight, and hence complement the security proofs of the designers. In particular, we show how to distinguish the CMC, EME, EME<sup>+</sup> and EME\* modes from random tweakable permutations with negligible effort and  $2^{n/2}$  chosen plaintexts, where  $n$  is the block size in bits. Further, we point out that both modes leak secret information via side-channel attacks (timing and power) due to the data-dependent internal multiplication operation.

**Keywords:** Block cipher, modes of operation, tweakable schemes, disk encryption, security bounds, distinguisher.

## 1 Introduction

A *block cipher* is defined as

$$E_K : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n, \quad (1)$$

where  $n$  denotes the block size, and  $k$  the length of the secret key,  $K$ . Meanwhile, a block cipher *mode of operation* is an encryption scheme,  $\mathbf{E}_K$  that makes use of a block cipher,  $E_K$  as a basic primitive in order to encrypt a message that is of length a multiple,  $m$  of the block size,  $n$ . Formally, this is defined as:

$$\mathbf{E}_K : \{0, 1\}^k \times \{0, 1\}^{mn} \rightarrow \{0, 1\}^{mn}, \quad (2)$$

where  $m$  denotes the number of  $n$ -bit blocks being operated upon.

---

\* The second author acknowledges the Malaysia IRPA grant (04-99-01-00003-EAR).

In Crypto '02, Liskov et al. [20] first proposed a new cryptographic primitive known as a *tweakable block cipher*. Informally, besides having  $n$ -bit plaintexts and a  $k$ -bit secret key,  $K$ , a tweakable block cipher also possesses an extra  $t$ -bit input, called a tweak,  $T$ . The tweakable block cipher,  $E_K^T$  is defined as follows:

$$E_K^T : \{0, 1\}^k \times \{0, 1\}^t \times \{0, 1\}^n \rightarrow \{0, 1\}^n. \quad (3)$$

Tweakable block ciphers are tailor-made for special applications such as disk-sector encryption, which requires the bulk encryption of large amounts of data for storage in different sectors on the disk. Therefore, tweakable block ciphers must fulfill the following properties:

- a single bit change in the plaintext should impact the complete ciphertext sector (variability).
- each sector can be encrypted independently of other sectors (efficiency).

A tweakable block cipher should be viewed as a strong random permutation and be indistinguishable from any other random tweakable permutation (an oracle that realizes a  $T$ -indexed family of random permutations and their inverses), even if an attacker has access to encryption and decryption oracles, ability to obtain adaptive-chosen plaintext/ciphertext queries, and control over the tweak.

Building on this construction, several *tweakable block cipher modes of operation*, namely CMC, EME, EME<sup>+</sup> and EME\* were recently presented in [9,10], [11,12], and [7,8] at Crypto '03, CT-RSA '04 and Indocrypt '04 respectively. Thus, there is continued interest not only by the designers but the crypto community in these modes and their variants. Formally, a tweakable block cipher mode of operation,  $\mathbf{E}_K^T$  is defined as:

$$\mathbf{E}_K^T : \{0, 1\}^k \times \{0, 1\}^t \times \{0, 1\}^{mn} \rightarrow \{0, 1\}^{mn}. \quad (4)$$

In this paper, we describe how to distinguish the CMC, EME, EME<sup>+</sup> and EME\* modes from random tweakable permutations, by using  $2^{n/2}$  *chosen plaintext (CP)* queries. Note that even with such queries, one would not be able to distinguish the underlying block cipher,  $E_K$  from random permutation.

Our results show that the upper bounds in the security proofs of the modes are tight. These complement the security proofs of the designers (up to a factor involving the number of blocks,  $m$ ), and in this respect are somewhat similar to those by Ferguson [5] on OCB, and by Mitchell [24] on XCBC, TMAC and OMAC, respectively. However, in our case it is interesting when comparing our results with the security proofs of the CMC, EME and EME\* in [9,10], [11,12], and [7,8]. The upper bounds on the advantage of an attacker for CMC, EME and EME\* are  $O((mq)^2 \times 2^{-n})$ ,  $O(q^2 \times 2^{-n})$  and  $O((m+q)^2 \times 2^{-n})$ , respectively, where  $q$  is the number of text queries made by the attacker. In our case,  $q = 2^{n/2}$  and our results are in line with the upper bound for EME but for CMC and EME\*, our results show that the upper bound does not need to depend on the number of blocks,  $m$  of the plaintext message. As an aside, we also discuss in the

Appendix how these modes leak secret information via the timing [18] and power [19] side-channels due to the data-dependent internal multiplication operation being used in them.

### 1.1 Related Work

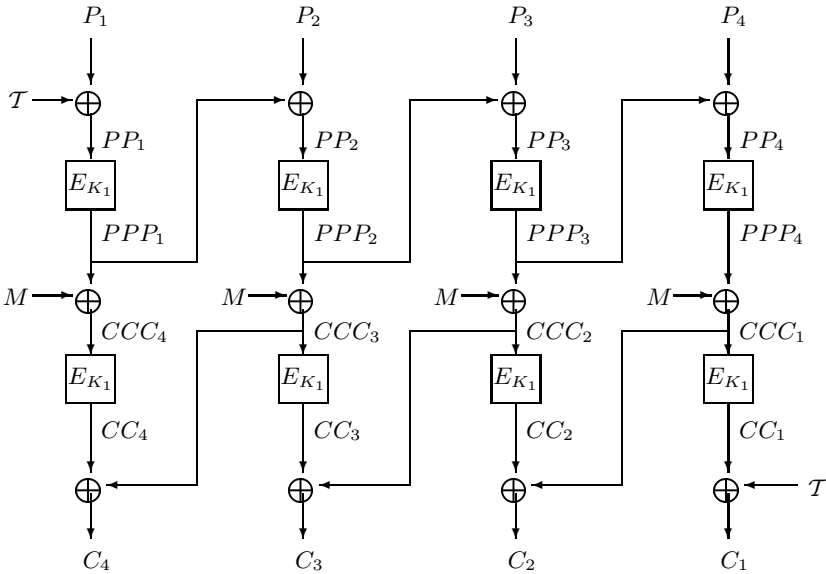
Tweakable block ciphers were first suggested by Liskov et al. [20] in 2002. Rogaway [25] proposed in 2002 the Encrypt-Mask-Decrypt (EMD), and Encrypt-Mask-Encrypt (EME) modes of operation, both of which are tweakable schemes. [25] was in fact the paper that proposed the early versions of the CMC and EME modes, then known as EMD and EME respectively. However, Joux [15] showed in 2003 that both these early modes are insecure and fall to simple distinguishing attacks. In particular, they could be distinguished with only 3 *adaptively-chosen plaintexts/ciphertexts (ACPs)*. Subsequently, Rogaway in collaboration with Halevi proposed new versions called CMC [9,10] and EME [11,12] respectively. To settle any further confusion, we emphasize here that though EMD was renamed to CMC, EME remained with the same name. For the rest of this manuscript, the term “EME” shall refer to the new EME version proposed in [11,12]. Finally, Halevi on his own recently proposed EME\* [7,8] as another variant of EME.

## 2 The CMC, EME, EME<sup>+</sup> and EME\* Modes

The CBC-Mask-CBC (CMC) mode of operation was presented by Halevi and Rogaway [9,10]. It operates on a plaintext message,  $P$  of  $m$  blocks, each block of  $n$  bits, i.e.  $P = (P_1, \dots, P_m)$  where  $|P_i| = n$ , for  $i \in \{1, \dots, m\}$ . Besides the input  $P$ , there is also another  $n$ -bit input tweak,  $T$ . The CMC uses a secret key,

**Table 1.** CMC Mode Encryption

<p><b>Algorithm <math>E_K^T(P_1 \dots P_m)</math></b>  <math>T = E_{K_2}(T)</math>  <math>PPP_0 = T</math>  <b>for</b> <math>i \leftarrow 1</math> <b>to</b> <math>m</math> <b>do</b>  <math>PP_i \leftarrow P_i \oplus PPP_{i-1}</math>  <math>PPP_i \leftarrow E_{K_1}(PP_i)</math>  <math>M \leftarrow 2(PPP_1 \oplus PPP_m)</math>  <b>for</b> <math>i \in [1 \dots m]</math> <b>do</b>  <math>CCC_i \leftarrow PPP_{m+1-i} \oplus M</math>  <math>CCC_0 \leftarrow 0^n</math>  <b>for</b> <math>i \in [1 \dots m]</math> <b>do</b>  <math>CC_i \leftarrow E_{K_1}(CCC_i)</math>  <math>C_i \leftarrow CC_i \oplus CCC_{i-1}</math>  <math>C_1 \leftarrow C_1 \oplus T</math>  <b>return</b> <math>C_1 \dots C_m</math></p>
---



**Fig. 1.** Encrypting under the CMC mode for a message of  $m = 4$  blocks, and where  $M = 2(PPP_1 \oplus PPP_4) = 2(CCC_1 \oplus CCC_4)$ , and  $T = E_{K_2}(T)$

$K = (K_1, K_2)$  where  $K_1$  is used for encryption of plaintext blocks while  $K_2$  is used to encrypt the tweak,  $T$ .

Table 1 gives the algorithm for CMC mode encryption<sup>1</sup>, while Figure 1 illustrates this for  $m = 4$  blocks. For all modes we discuss in this paper, multiplication by 2 is done in  $\text{GF}(2^n)$ . More details of this are in the Appendix and [9,10]. Note that the output ciphertext blocks in CMC are taken in reverse order from the input plaintext blocks. This was done by the designers to maintain the symmetry of the encryption and decryption operations.

Further, note how similar the CMC mode is to the double modes in particular double CBC [13]. Both require two encryption layers to process each block of plaintext. The main differences are that CMC uses an extra internal mask,  $M$ , and only one key is used in the CMC’s block encryptions, in contrast to double modes where an independent key is used for each different block encryption layer. A further difference between them is that in the CMC mode, the second layer of encryption is reversed, i.e. performed right-to-left rather than left-to-right. That being said, we remark that aside from the single key used for mode encryption, CMC does use another key to encrypt the tweak. Therefore, both CMC and double modes effectively use two keys. In terms of efficiency, CBC requires  $2m$  single  $E_K$  encryptions to process an  $m$ -block plaintext, while CMC requires  $2m + 1$  single  $E_K$  encryptions, where one encryption is due to the computation of  $T$ .

<sup>1</sup> We have omitted describing the mode decryptions as they are irrelevant to our attacks. The interested reader is referred to [9,10] for further details.



Meanwhile, Halevi and Rogaway [11,12] also presented another mode of operation, namely the the Encrypt-Mask-Encrypt (EME). The EME mode encryption is defined in Table 2, while Figure 2 illustrates this for  $m = 4$  blocks.

Observe that the EME is essentially a double ECB mode [13] except for the use of extra XOR maskings in between the block encryption layers. While a double ECB mode would use two unique keys for each encryption layer, EME only uses one key, and so the keylength of EME is only half that of the double modes. Further comparing efficiency, double modes would require  $2m$  single  $E_K$  encryptions to process an  $m$ -block plaintext, while EME requires  $2m + 2$  single  $E_K$  encryptions, where one extra encryption is due to the computation of  $L$  while the other is due to the computation of  $MC = E_K(MP)$ .

Two other variants of the EME are the EME<sup>+</sup> and EME\*, proposed respectively in [11,12] and [7,8] for handling long plaintext messages with blocks,  $m > n$ .

**Table 2.** EME Mode Encryption

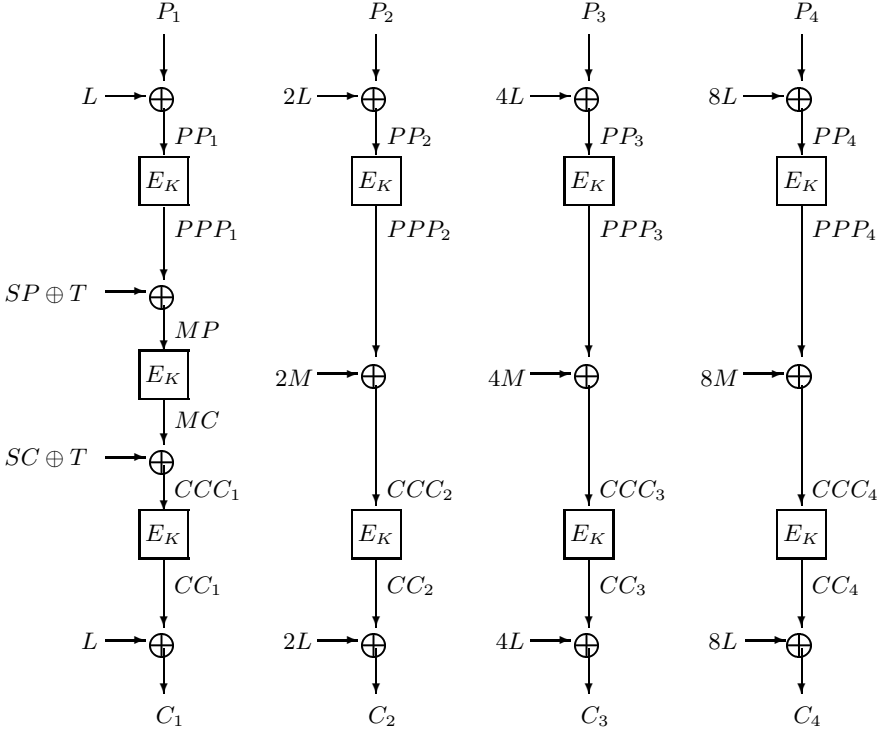
<b>Algorithm</b> $E_K^T(P_1 \dots P_m)$
$L = 2E_K(0^n)$
<b>for</b> $i \in [1 \dots m]$ <b>do</b>
$PP_i \leftarrow 2^{i-1}L \oplus P_i$
$PPP_i \leftarrow E_K(PP_i)$
$SP \leftarrow PPP_2 \oplus \dots \oplus PPP_m$
$MP \leftarrow PPP_1 \oplus SP \oplus T$
$MC \leftarrow E_K(MP)$
$M \leftarrow MP \oplus MC$
<b>for</b> $i \in [2 \dots m]$ <b>do</b>
$CCC_i \leftarrow PPP_i \oplus 2^{i-1}M$
$SC \leftarrow CCC_2 \oplus \dots \oplus CCC_m$
$CCC_1 \leftarrow MC \oplus SC \oplus T$
<b>for</b> $i \in [1 \dots m]$ <b>do</b>
$CC_i \leftarrow E_K(CCC_i)$
$C_i \leftarrow CC_i \oplus 2^{i-1}L$
<b>return</b> $C_1 \dots C_m$

### 3 The Security Bounds for CMC, EME, EME<sup>+</sup> and EME\* Are Tight

In this section, we show that the security bounds for the CMC, EME, EME<sup>+</sup> and EME\* modes are tight by illustrating how to distinguish these modes from random tweakable permutations.

#### 3.1 The CMC Mode

We start with a basic distinguisher. Obtain the encryptions of  $2^{n/2}$  messages,  $P^{(i)} = (i, 0, \dots, 0)$  under tweaks,  $T_j = j$  where  $i, j \in \{1, 2, \dots, 2^{n/2}\}$  and thus



**Fig. 2.** Encrypting under the EME mode for a message of  $m = 4$  blocks, where  $L = 2E_K(0^n)$ ,  $SP = PPP_2 \oplus PPP_3 \oplus PPP_4$ ,  $M = MP \oplus MC$ , and  $SC = CCC_2 \oplus CCC_3 \oplus CCC_4$

obtain  $2^n$  pairs  $(P^{(i)}, T_j), (P^{(i)'}, T'_j)$  such that if the black box contains the CMC, then at least one pair has a collision at  $PP_1$ , and therefore satisfies the conditions:

$$PPP_1 = E_K(PP_1) = E_K(P_1 \oplus E_{K_2}(T)) \quad (5)$$

$$= E_K(PP'_1) = E_K(P'_1 \oplus E_{K_2}(T')) \quad (6)$$

$$= PPP'_1. \quad (7)$$

This collision propagates through to the other blocks, and so

$$M = 2(PPP_1 \oplus PPP_m) \quad (8)$$

$$= 2(PPP'_1 \oplus PPP_m) \quad (9)$$

$$= M'. \quad (10)$$

Since all the plaintext blocks in the two messages are equal except for the first block, and since the only other block affected by the tweak,  $T$  is the last block, then we will observe

$$C_i = C'_i ; i = 2, \dots, m. \quad (11)$$

Checking for this is trivial.

In contrast for a random tweakable permutation, such a condition would only occur with probability  $2^{-n \times (m-1)}$ , whereas for CMC, this is satisfied with probability  $2^{-n}$ . Therefore, one could distinguish between CMC and a random tweakable permutation.

This allows to distinguish the CMC with  $2^n$  chosen plaintexts/tweaks and negligible effort.

We can do better by noting that a collision at  $PP_1$  works as long as one finds  $P, P', T, T'$  such that

$$P_1 \oplus P'_1 = T \oplus T'. \tag{12}$$

By birthday analysis, one should be able to come up with this after seeing about  $2^{n/2}$   $P, P'$  pairs, i.e. a collision occurs. Note that equation (12) is essentially random since a block cipher primitive was applied to derive these values. In all, this involves about  $2^{n/2}$  chosen plaintexts/tweaks.

### 3.2 The EME, EME<sup>+</sup> and EME\*

Distinguishing the EME mode works by obtaining the encryptions of  $2^{n/2}$  messages  $P^{(i)} = (P_1^{(i)}, P_2^{(i)}, P_3^{(i)}, \dots, P_m^{(i)}) = (i, i, P_3^{(i)}, \dots, P_m^{(i)})$  for  $i \in \{1, 2, \dots, 2^{n/2}\}$ , where  $P_j^{(i)}$  for  $j \in \{3, \dots, m\}$  are any constant fixed value<sup>2</sup>. These form  $2^n$  pairs of  $P, P'$  that differ only in their first two blocks and equal in all other blocks.

For any such pair, if the black box contains the EME, then with a probability  $2^{-n}$  a collision occurs at their  $MP$  point, that is

$$PPP_1 \oplus SP \oplus T = PPP'_1 \oplus SP' \oplus T \tag{13}$$

and so

$$MP = MP'. \tag{14}$$

When this happens, then

$$MC = MC', \tag{15}$$

$$M = MP \oplus MC = MP' \oplus MC' = M' \tag{16}$$

and this collision propagates to the other blocks via  $M$ . Since all the blocks in the two messages are equal except for the first two blocks, then we will observe

$$C_i = C'_i ; i = 3, \dots, m. \tag{17}$$

Note that for a random tweakable permutation, such a condition would only occur with probability  $2^{-n \times (m-2)}$ ,  $m > 2$ , whereas for EME, this is satisfied with

---

<sup>2</sup> In fact, we can use any group of two messages as long as they are distinct in any two blocks and equal in the other blocks.

probability  $2^{-n}$ . Therefore, one could distinguish between EME and a random tweakable permutation.

In summary, we can distinguish the EME with  $2^{n/2}$  chosen plaintexts and negligible effort.

Note that this distinguisher equally applies to both EME<sup>+</sup> and EME\*, two very similar variants of EME proposed respectively in [11,12] and [7,8] for handling long plaintext messages with blocks,  $m > n$ . In this case, a similar collision occurs between any such pair with probability  $2^{-n}$  at  $MP_1$  halfway through encrypting their first blocks, propagating further to  $MC_1$  and hence  $M_1 = M'_1$  and this propagates through to all other blocks, and the distinguishing attack proceeds along the same lines as that on EME.

## 4 Concluding Remarks

We have shown that the upper bounds of the security proofs of the CMC, EME, EME<sup>+</sup> and EME\* modes are tight, in complement to the designers' results up to a factor involving the number of blocks. We list in Table 3 a comparison between our results and previous ones on the early EMD and old EME modes. Note that the complexities are given as the number of texts required as well as the number of single encryptions. We also remark that Joux's results on the earlier modes require adaptively-chosen plaintext/ciphertext (*ACP*) queries, while ours require the more practical chosen (*CP*) plaintext queries. Further, Joux's result on the old EME requires the attacker to have control over the tweak,  $T$  and hence control over usage of different sectors. In contrast, our results on the new EME in Section 3.2 makes use of the same  $T$  and this can be carried out on the same disk sector.

**Table 3.** Comparing Distinguisher Complexities on EMD, CMC, EME, EME<sup>+</sup> and EME\*

Mode	Text Complexities	Source
EMD [25]	3 <i>ACPs</i>	[15]
Old EME [25]	3 <i>ACPs</i>	[15]
CMC [9,10]	$2^{n/2}$ <i>CPs</i>	This paper
New EME [11,12]	$2^{n/2}$ <i>CPs</i>	This paper
EME <sup>+</sup> [11,12]	$2^{n/2}$ <i>CPs</i>	This paper
EME* [7,8]	$2^{n/2}$ <i>CPs</i>	This paper

The upper bounds on the advantage of an attacker for CMC[9,10], EME [11,12], and EME\* [7,8] are  $O((mq)^2 \times 2^{-n})$ ,  $O(q^2 \times 2^{-n})$  and  $O((m+q)^2 \times 2^{-n})$ , respectively. Since  $q = 2^{n/2}$  in our case, our results match the upper bound for EME but show that the upper bounds of CMC and EME\* do not need to depend on the number of blocks,  $m$ .

## Acknowledgement

We thank the (past and present) anonymous referees for their constructive comments that have greatly improved this paper. We thank God for His blessings, and our wives and daughters for their unwavering support.

## References

1. E. Biham, “Cryptanalysis of Multiple Modes of Operation”, *Asiacrypt '94, Lecture Notes in Computer Science*, Vol. 917, pp. 278–292, Springer-Verlag, 1994.
2. E. Biham, “Cryptanalysis of Multiple Modes of Operation”, *Journal of Cryptology*, Vol. 11, pp. 45–58, Springer-Verlag, 1998.
3. E. Biham, “Cryptanalysis of Triple Modes of Operation”, *Journal of Cryptology*, Vol. 12, pp. 161–184, Springer-Verlag, 1999.
4. J. Black and P. Rogaway, “A Block-cipher Mode of Operation for Parallizable Message Authentication”, *Advances in Cryptology - Eurocrypt '02, Lecture Notes in Computer Science*, Vol. 2332, pp. 384–397, Springer-Verlag, 2002.
5. N. Ferguson, “Collision Attacks on OCB”. Comments to NIST, February 11, 2002. Available from NIST’s web page at <http://csrc.nist.gov/CryptoToolkit/modes/>.
6. FIPS 81, “DES Modes of Operation”, US Department of Commerce, National Bureau of Standards, 1980.
7. S. Halevi, “EME\*: Extending EME to Handle Arbitrary-length Messages with Associated Data”, *Progress in Cryptology - Indocrypt '04, Lecture Notes in Computer Science*, Vol. 3348, pp. 315–327, Springer-Verlag, 2004.
8. S. Halevi, “EME\*: Extending EME to Handle Arbitrary-length Messages with Associated Data”, full version, *Cryptology ePrint archive*, <http://eprint.iacr.org/2004/125/>, 2004.
9. S. Halevi and P. Rogaway, “A Tweakable Enciphering Mode”, *Advances in Cryptology - Crypto '03, Lecture Notes in Computer Science*, Vol. 2729, pp. 482–499, Springer-Verlag, 2003.
10. S. Halevi and P. Rogaway, “A Tweakable Enciphering Mode”, full version, *Cryptology ePrint archive*, <http://eprint.iacr.org/2003/148/>, 2003.
11. S. Halevi and P. Rogaway, “A Parallelizable Enciphering Mode”, *Topics in Cryptology - CT-RSA '04, Lecture Notes in Computer Science*, Vol. 2964, pp. 292–304, Springer-Verlag, 2004.
12. S. Halevi and P. Rogaway, “A Parallelizable Enciphering Mode”, full version, *Cryptology ePrint archive*, <http://eprint.iacr.org/2003/147/>, 2003.
13. H. Handschuh and B. Preneel, “On the Security of Double and 2-key Triple Modes of Operation”, *FSE '99, Lecture Notes in Computer Science*, Vol. 1636, pp. 215–230, Springer-Verlag, 1999.
14. T. Iwata, “Comments on “On the Security of XCBC, TMAC and OMAC” by Mitchell”. Comments to NIST, September 19, 2003. Available from NIST’s web page at <http://csrc.nist.gov/CryptoToolkit/modes/>.
15. A. Joux, “Cryptanalysis of the EMD Mode of Operation”, *Advances in Cryptology - Eurocrypt '03, Lecture Notes in Computer Science*, Vol. 2656, pp. 1–16, Springer-Verlag, 2003.
16. J. Kilian and P. Rogaway, “How to Protect DES Against Exhaustive Key Search”, *Advances in Cryptology - Crypto '96, Lecture Notes in Computer Science*, Vol. 1109, pp. 252–267, Springer-Verlag, 1996.

17. J. Kilian and P. Rogaway, “How to Protect DES Against Exhaustive Key Search (an Analysis of DESX)”, *Journal of Cryptology*, Vol. 14, No.1, pp. 17–35, 2001.
18. P. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”, *Advances in Cryptology - Crypto '96*, Lecture Notes in Computer Science, Vol. 1109, pp. 104–113, Springer-Verlag, 1996.
19. P. Kocher, J. Jaffe and B. Jun, “Differential Power Analysis,” *Advances in Cryptology - Crypto '99*, Lecture Notes in Computer Science, Vol. 1666, pp. 388–397, Springer-Verlag, 1999.
20. M. Liskov, R. Rivest and D. Wagner, “Tweakable Block Ciphers”, *Advances in Cryptology - Crypto '02*, Lecture Notes in Computer Science, Vol. 2442, pp. 31–46, Springer-Verlag, 2002.
21. S. Mangard, “A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion”, *ICISC '02*, Lecture Notes in Computer Science, Vol. 2587, pp. 343–358, Springer-Verlag, 2003.
22. S. Mangard, “Securing Implementations of Block Ciphers against Side-Channel Attacks”, PhD Dissertation, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, August 2004.
23. R. Mayer-Sommer, “Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards”, *CHES '00*, Lecture Notes in Computer Science, Vol. 1965, pp. 78–92, Springer-Verlag, 2000.
24. C.J. Mitchell, “On the Security of XCBC, TMAC and OMAC”. Comments to NIST, August 19, 2003. Available from NIST’s web page at <http://csrc.nist.gov/CryptoToolkit/modes/>.
25. P. Rogaway, “The EMD Mode of Operation (a Tweaked, Wide-blocksize, Strong PRP)”, *Cryptology ePrint archive*, <http://eprint.iacr.org/2002/148/>, 2002.

## Appendix: Timing and Power Attacks on CMC, EME, EME<sup>+</sup> and EME\*

The multiply-by-2 operation used in CMC [9,10], EME [11,12], EME<sup>+</sup> and EME\* [7,8] is data-dependent, in the sense that a conditional XOR is performed dependent on the value of the most significant bit (MSB) of the input data. In more detail, a multiplication of a 128-bit  $L = L_{n-1} \dots L_1 L_0 \in \{0, 1\}^n$  by 2 is defined by the designers as:

$$2L = L \ll 1 \quad \text{if MSB}(L) = 0, \quad (18)$$

and

$$2L = L \ll 1 \oplus \text{Const87} \quad \text{if MSB}(L) = 1, \quad (19)$$

where  $L \ll 1$  means  $L_{n-2}L_{n-3} \dots L_1L_00$ , and **Const87** means a constant decimal 87.

This conditional XOR operation allows for one to mount a timing attack [18] on these modes. In particular, the operation would take a slightly longer time if the MSB of  $L$  is 1, compared to the case where it is 0. This allows an attacker to differentiate between the MSB values of the input,  $L$  to the multiply-by-2 operation. In fact, since the modes are claimed to be secure against variable length input queries, an attacker could obtain queries from the modes under differing input lengths, each time increasing the block length by 1.

For example, suppose we have  $m = 2$  for EME. We input a plaintext  $P = (P_1, P_2)$  to the EME oracle. Consider the processing of the second<sup>3</sup> block,  $P_2$  which involves two  $2L$  operations and one  $2M$  operation. Depending on the values of  $L$  and  $M$ , the processing of the second block would require either of 4 cases: two conditional XOR operations (due to an MSB of 1 in the input to  $2L$ ), one conditional XOR operation (due to an MSB of 1 in the input to  $2M$ ), all three XORs or neither. This therefore leaks information on the MSBs of  $L$  and  $M$ . Further, repeating the query by reusing the same plaintext but with an extra block ( $m = 3$  in this case), to obtain  $P = (P_1, P_2, P_3)$  then this extra block would entail two additional  $2L$  and one additional  $2M$  operation. Similarly, observing the difference in time taken to process this plaintext, one could guess which of the  $2L$  and  $2M$  operations on the block  $P_3$  require the conditional XOR operation, and guess the MSBs of their inputs.

**Table 4.** Four Cases of Conditional XORs at Each Block

Total XORs	MSB of input to $2L$	MSB of input to $2M$
0	0	0
1	0	1
2	1	0
3	1	1

For each block we have 4 cases: one conditional XOR, two, three or none, then since we are accumulating on all blocks, the sum of the differences (between whether an XOR is occurring at each multiply-by-2 operation) would be bigger than the case where only one block is considered. And because each of the 4 cases is due to a unique cause (as per Table 4), their combinations (after accumulation) are distinguishable since they do not run through the entire combinations space.

Note that this leakage of the MSBs of  $L$  and  $M$  could also be observed via power analysis [19]. A conditional XOR with `Const87 = 10000111B` (in binary) means 4 bits will be complemented hence a state-changing transition (from 0 to 1 or from 1 to 0) would occur. It has been noted in [22,21] that state-changing transitions dissipate significantly more power than state-preserving (from 0 to 0 or from 1 to 1) transitions, thus an XOR occurring for each multiply-by-2 operation would mean significantly more power dissipation than the case where the XOR is not done, especially since we would be accumulating on all blocks with 3 conditional XORs per block.

---

<sup>3</sup> The first block contains only the  $L$  computation, which has a fixed input of  $0^n$  and therefore MSB of 0, so no conditional XOR is performed.

# On the Security of Encryption Modes of MD4, MD5 and HAVAL\*

## (Extended Abstract)

Jongsung Kim<sup>1,\*\*</sup>, Alex Biryukov<sup>1</sup>, Bart Preneel<sup>1</sup>, and Sangjin Lee<sup>2</sup>

<sup>1</sup> Katholieke Universiteit Leuven, ESAT/SCD-COSIC,  
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium  
{Kim.Jongsung, Alex.Biryukov, Bart.Preneel}@esat.kuleuven.be

<sup>2</sup> Center for Information Security Technologies(CIST),  
Korea University, Seoul, Korea  
{sangjin}@cist.korea.ac.kr

**Abstract.** In this paper, we cryptanalyze the compression functions of MD4, MD5 and 4-, 5-pass HAVAL in encryption mode. We exploit the recently proposed related-key rectangle and boomerang techniques to show non-randomness of MD4, MD5 and 4-, 5-pass HAVAL and to distinguish them from a randomly chosen cipher. The attacks are highly practical and have been confirmed by our experiments.

## 1 Introduction

MD4 [14] is a cryptographic hash function introduced in 1990 by Rivest. It uses basic arithmetic operations and several Boolean functions which are suitable for fast software implementations on 32-bit processors. After MD4 was published, several hash functions based on the design philosophy of MD4 have been proposed: MD5 [15], HAVAL [23], RIPEMD [24], RIPEMD-160 [5], SHA-1 [25], SHA-256 [26], etc.

In 2004 and 2005 several important cryptanalytic articles [1, 2, 18, 19, 20, 21] have been published that demonstrate collisions for the MD4-family of hash functions. Especially, a “precise” differential attack proposed by Wang et al.

---

\* This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme under Contract IST2002507932 ECRYPT and in part by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

\*\* The first author was financed by a Ph.D. grant of the Katholieke Universiteit Leuven and supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (KRF-2005-213-D00077).



**Table 1.** Distinguishing Attacks of Encryption Modes of MD4, MD5 and HAVAL

Primitive	Type of Attack	Number of Source Keys	Data Complexity	Number of Weak Keys	Paper
MD4	R	2	$2^{69}$ RK-CP	.	This paper
	$B^\dagger$	2	$2^{18}$ RK-CP/ $2^{18}$ RK-ACC	.	This paper
	$B^\dagger$	2	2RK-CP/2RK-ACC	$2^{320}$	This paper
	R	4	$2^{69}$ RK-CP	.	This paper
	$B^\dagger$	4	$2^6$ RK-CP/ $2^6$ RK-ACC	.	This paper
	$B^\dagger$	4	2RK-CP/2RK-ACC	$2^{384}$	This paper
MD5	D	1	$2^{50}$ CP	.	[16]
	R	2	$2^{102.8}$ RK-CP	.	This paper
	B	2	$2^{80.6}$ RK-CP/ $2^{78.6}$ RK-ACC	.	This paper
	$B^\dagger$	2	12RK-CP/12RK-ACC	$2^{96}$	This paper
	R	4	$2^{71.1}$ RK-CP	.	This paper
	$B^\dagger$	4	$2^{13.6}$ RK-CP/ $2^{11.6}$ RK-ACC	.	This paper
HAVAL (4 passes)	D	1	$2^{127}$ CP	.	[22]
	R	2	$2^{148.5}$ RK-CP	.	This paper
	$B^\dagger$	2	$2^{37.9}$ RK-CP/ $2^{35.9}$ RK-ACC	.	This paper
	$B^\dagger$	2	$2^{12.3}$ RK-CP/ $2^{12.3}$ RK-ACC	$2^{576}$	This paper
	R	4	$2^{133}$ RK-CP	.	This paper
	$B^\dagger$	4	$2^{11.6}$ RK-CP/ $2^{9.6}$ RK-ACC	.	This paper
HAVAL (5 passes)	D	1	$2^{170}$ CP	.	[22]
	R	2	$2^{188.6}$ RK-CP	.	This paper
	B	2	$2^{127.9}$ RK-CP/ $2^{125.9}$ RK-ACC	.	This paper
	R	4	$2^{158.5}$ RK-CP	.	This paper
	B	4	$2^{63}$ RK-CP/ $2^{61}$ RK-ACC	.	This paper

$^\dagger$ : the attack can be implemented in a real time

D: Differential, B: Boomerang, R: Rectangle

RK: Related-Key, CP: Chosen Plaintexts, ACC: Adaptively Chosen Ciphertexts

Time complexity is the same as the amount of data complexity

enables us to greatly improve previous known collision attacks of MD4, MD5, HAVAL, RIPEMD, SHA-0 and SHA-1 [18, 19, 20, 21].

There have been also several cryptanalytic articles that investigate non-randomness of the compression functions of MD5, HAVAL, SHA-1 and SHA-256 in encryption mode. The encryption modes of SHA-1 and SHA-256 have been proposed in the NESSIE project, which are called SHACAL-1 and SHACAL-2 [7], respectively. For the encryption modes of SHA-1 and SHA-256, the security has been checked against various block cipher cryptanalyses [3, 6, 8, 9, 10, 12, 13, 16, 17], while differential cryptanalysis has been applied to the encryption modes of MD5 and HAVAL [16, 22].

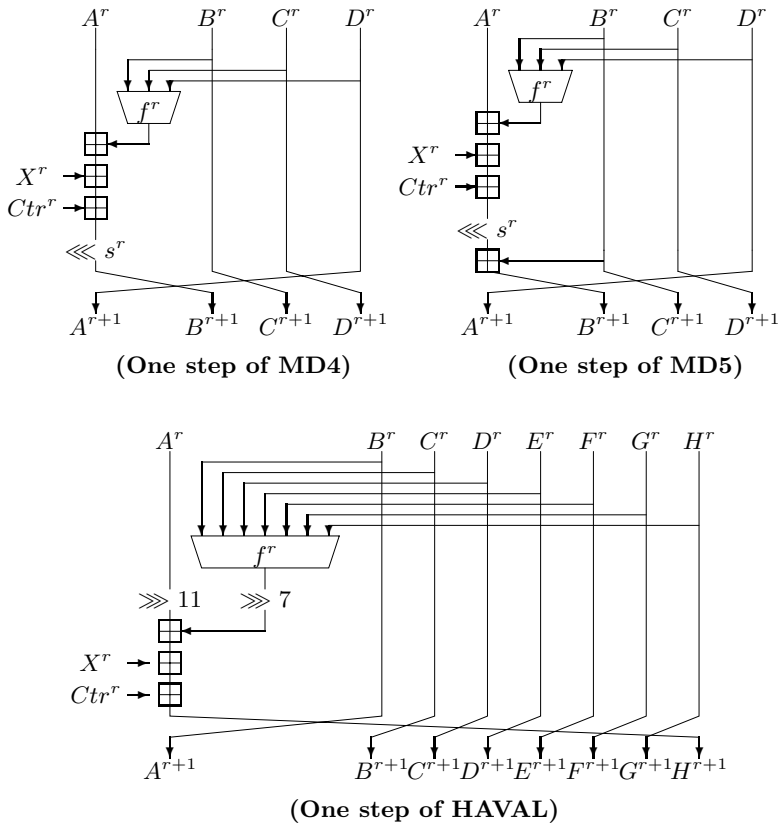
In this paper, we check the security of encryption modes of MD4, MD5 and HAVAL against the recently proposed related-key rectangle and boomerang attacks [4, 9, 10, 13], and we compare our results with the previous ones in terms of distinguishing attacks. Especially, we can distinguish the encryption modes of MD4, MD5 and 4-pass HAVAL from a randomly chosen cipher in practice by using a related-key boomerang attack. Furthermore, we can distinguish them more efficiently for a large class of weak keys (i.e., special subset of messages in hash mode). See Table 1 for a summary of our results and a comparison with the previous attacks.

## 2 Description of MD4, MD5 and HAVAL

The MD4, MD5 and HAVAL hash functions are message digest algorithms which compress any arbitrary-bit length message into a hash value with a small and fixed bit-length. These hash functions are performed based on the well-known Davies-Meyer construction, which is described as follows. Before applying the hash function to a message  $M$  of arbitrary bit-length, it is divided into  $l$ -bit sub-messages  $M_0, M_1, \dots, M_{n-1}$ , where  $l$  is specified. Then the  $t$ -bit hash value  $I_n$  for the message  $M$  is computed as follows:

**Table 2.** Parameters of MD4, MD5 and HAVAL

Hash Functions	Bit-Length of Message Block ( $l$ )	Bit-Length of Hash Value ( $t$ )	# of Passes	# of Steps in a Pass	Total # of Steps
MD4	512	128	3	16	48
MD5	512	128	4	16	64
HAVAL	1024	256	3,4 or 5	32	96, 128 or 160



**Fig. 1.** The  $r$ -th Step Functions of MD4, MD5 and HAVAL

$$I_0 = IV; I_{i+1} = \mathbf{com}(I_i, M_i) = E(I_i, M_i) + I_i \text{ for } 0 \leq i < n, \quad (1)$$

where  $IV$  is a  $t$ -bit fixed initial value,  $\mathbf{com}$  is a compression function and  $E$  is an iterative step function. In MD4, MD5 and HAVAL, the function  $E$  is composed of 3, 4 or 5 passes and in each pass there are 16 or 32 steps that use only simple basic operations and Boolean functions on 32-bit words. The  $t$ -bit input  $I_i$  is loaded into  $t/32$  32-bit registers denoted  $(A^0, B^0, \dots)$  and the  $l$ -bit message block is divided into  $l/32$  32-bit words denoted  $(X^0, X^1, \dots, X^{l/32})$ . The  $t/32$  registers are updated through a number of steps. In each pass, every message word  $X^i$  is used exactly once in a specified order, and a fixed Boolean function  $f$  and 32-bit constants  $Ctr$  are used. Table 2 shows the parameters of MD4, MD5 and HAVAL, and Fig. 1 shows the  $r$ -th step of MD4, MD5 and HAVAL. In Fig. 1, the rotation amount  $s^r$  is specified. See [14, 15, 23] for details.

Each of the steps described in Fig. 1 is an invertible function for each message word  $X^r$ . Hence, if we insert a secret key in the message part of  $M_i$  and a plaintext in the chaining value part of  $I_i$ , we get an invertible function from a compression function by removing the final addition with the previous chaining value. That is,  $E(I_i, M_i)$  of Eq. (1) can be used in encryption mode  $E(P, K)$ , where  $P$  is a plaintext and  $K$  is a secret key. In the encryption modes of MD4, MD5 and HAVAL, we use the terminology *rounds* instead of *steps* and we use the notation  $P$  and  $K$  for a plaintext and a key, respectively.

### 3 Related-Key Rectangle and Boomerang Attacks

Related-key rectangle and boomerang attacks were presented in several papers [4, 9, 10, 13]. They exploit related-key rectangle and boomerang distinguishers based on 2, 4 or 256 related keys. In this paper, we use related-key rectangle and boomerang distinguishers based on 2 or 4 related keys.

The following notations are used to facilitate the descriptions of related-key rectangle and boomerang distinguishers.

- $E : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  : a block cipher that uses  $\{0, 1\}^k$  and  $\{0, 1\}^n$  as key space and plaintext/ciphertext space, respectively.
- $E = E^1 \circ E^0$  (i.e.,  $E_K(P) = E_K^1 \circ E_K^0(P)$ ) :  $E$  is composed of  $E^0$  and  $E^1$  ( $E$  first performs  $E^0$  and then  $E^1$ ), where  $K$  is a master key and  $P$  is a plaintext.
- $p(\alpha, \beta, \Delta K)$  : a probability of a related-key differential  $\alpha \rightarrow \beta$  for  $E^0$  under the related-key difference  $\Delta K$ , i.e.,  $p(\alpha, \beta, \Delta K) = \Pr_{X, K}[E_K^0(X) \oplus E_{K \oplus \Delta K}^0(X \oplus \alpha) = \beta]$ . Note that this is same as a probability of a related-key differential  $\beta \rightarrow \alpha$  for  $(E^0)^{-1}$  under the related-key difference  $\Delta K$ .
- $q(\gamma, \delta, \Delta K)$  : a probability of a related-key differential  $\gamma \rightarrow \delta$  for  $E^1$  under the related-key difference  $\Delta K$ , i.e.,  $q(\gamma, \delta, \Delta K) = \Pr_{X, K}[E_K^1(X) \oplus E_{K \oplus \Delta K}^1(X \oplus \gamma) = \delta]$ . Note that this is same as a probability of a related-key differential  $\delta \rightarrow \gamma$  for  $(E^1)^{-1}$  under the related-key difference  $\Delta K$ .

- $p(D, \beta, \Delta K)$  : a probability of a related-key truncated differential  $\beta \rightarrow \alpha'$  for  $(E^0)^{-1}$  under the related-key difference  $\Delta K$ , where  $D$  is a nonempty set and  $\alpha' \in D$ , i.e.,  $p(D, \beta, \Delta K) = \Pr_{X,K}[(E_K^0)^{-1}(X) \oplus (E_{K \oplus \Delta K}^0)^{-1}(X \oplus \beta) \in D]$ .
- $q(\gamma, D, \Delta K)$  : a probability of a related-key truncated differential  $\gamma \rightarrow \delta'$  for  $E^1$  under the related-key difference  $\Delta K$ , where  $D$  is a nonempty set and  $\delta' \in D$ , i.e.,  $q(\gamma, D, \Delta K) = \Pr_{X,K}[E_K^1(X) \oplus E_{K \oplus \Delta K}^1(X \oplus \gamma) \in D]$ .

We first describe a related-key rectangle distinguisher based on two related keys. The related-key rectangle distinguisher works in the following process.

- Choose two plaintexts  $P_0$  and  $P_1^*$  at random and compute two other plaintexts  $P_0^* = P_0 \oplus \alpha$  and  $P_1 = P_1^* \oplus \alpha$ .
- With a chosen plaintext attack, obtain the corresponding ciphertexts  $C_0 = E_K(P_0)$ ,  $C_1 = E_K(P_1)$ ,  $C_0^* = E_{K^*}(P_0^*)$  and  $C_1^* = E_{K^*}(P_1^*)$ , where  $K \oplus K^* = \Delta K$ .
- Check if  $C_0 \oplus C_1^*$ ,  $C_0^* \oplus C_1 \in D$ .

What is the probability that the ciphertext quartet satisfies the last  $D$  test? The probability is computed as follows. Let  $X_0, X_1, X_0^*$  and  $X_1^*$  denote the encrypted values of  $P_0, P_1, P_0^*$  and  $P_1^*$  under  $E^0$ , respectively. Then the probabilities that  $X_0 \oplus X_0^* = \beta$  and  $X_1 \oplus X_1^* = \beta'$  are  $p(\alpha, \beta, \Delta K)$  and  $p(\alpha, \beta', \Delta K)$ , respectively. In the above process we randomly choose two plaintexts  $P_0$  and  $P_1^*$  and thus we expect  $X_0 \oplus X_1^* = \gamma$  with probability  $2^{-n}$ . Therefore, for any  $\beta, \beta'$  and  $\gamma$ ,  $X_0 \oplus X_0^* = \beta$ ,  $X_1 \oplus X_1^* = \beta'$  and  $X_0 \oplus X_1^* = \gamma$  (as in these cases  $X_0^* \oplus X_1 = (X_0 \oplus \beta) \oplus (X_1^* \oplus \beta') = \beta \oplus \beta' \oplus \gamma$ ) hold with probability  $p(\alpha, \beta, \Delta K) \cdot p(\alpha, \beta', \Delta K) \cdot 2^{-n}$ . Since the probabilities of related-key truncated differentials  $\gamma \rightarrow \delta' (\in D)$  and  $\beta \oplus \beta' \oplus \gamma \rightarrow \delta' (\in D)$  for  $E^1$  under related-key difference  $\Delta K$  are  $q(\gamma, D, \Delta K)$  and  $q(\gamma \oplus \beta \oplus \beta', D, \Delta K)$ , the probability that the last  $D$  test in the above process is satisfied equals

$$Pr[REC-2] = \sum_{\beta, \beta', \gamma} p(\alpha, \beta, \Delta K) \cdot p(\alpha, \beta', \Delta K) \cdot 2^{-n} \cdot q(\gamma, D, \Delta K) \cdot q(\gamma \oplus \beta \oplus \beta', D, \Delta K).$$

On the other hand, for a random cipher, the  $D$  test holds with probability  $|D|^2 \cdot 2^{-2n}$  and thus if the above probability is larger than  $|D|^2 \cdot 2^{-2n}$  for any triple  $(\alpha, D, \Delta K)$ , the related-key rectangle distinguisher based on two related keys can be used to distinguish  $E$  from a random cipher.

How many plaintext pairs are required to get at least two ciphertext quartets (this amount of quartets will be used in our attacks) that satisfy the  $D$  test? If the number of plaintext pairs  $(P_i, P_i^*)$  we collect is  $m$ , we can generate  $m^2 \cdot 2^{-1}$  quartets and thus we have at least  $m^2 \cdot 2^{-1} \cdot Pr[REC-2]$  ciphertext quartets which satisfy the  $D$  test. Therefore, in order to get at least 2 such quartets we need about  $4 \cdot (Pr[REC-2])^{-1/2}$  chosen plaintext queries. It means that the number of required plaintexts to use this distinguisher is at least  $2^{n/2}$ . However, under an adaptive chosen plaintext and ciphertext attack we can make a related-key

boomerang distinguisher which can remove the factor  $2^{n/2}$  in the data requirement. The related-key boomerang distinguisher based on two related keys works as follows.

- Choose two plaintexts  $P_0$  and  $P_0^*$  such that  $P_0 \oplus P_0^* = \alpha$ , and obtain the corresponding ciphertexts  $C_0 = E_K(P_0)$  and  $C_0^* = E_{K^*}(P_0^*)$ , where  $K \oplus K^* = \Delta K$ .
- Compute other two ciphertexts  $C_1 = C_0^* \oplus \delta$  and  $C_1^* = C_0 \oplus \delta$ , and obtain the corresponding plaintexts  $P_1 = E_K^{-1}(C_1)$  and  $P_1^* = E_{K^*}^{-1}(C_1^*)$ .
- Check  $P_1 \oplus P_1^* \in D$ .

Similarly, we can check the probability that the last  $\alpha'$  test is satisfied. The probability that  $X_0 \oplus X_0^* = \beta$  is  $p(\alpha, \beta, \Delta K)$  (in the encryption direction) and the probabilities that  $X_0^* \oplus X_1 = \gamma$  and  $X_0 \oplus X_1^* = \gamma'$  are  $q(\gamma, \delta, \Delta K)$  and  $q(\gamma', \delta, \Delta K)$  (in the decryption direction), respectively. Therefore, for any  $\beta, \gamma$  and  $\gamma'$ ,  $X_0 \oplus X_0^* = \beta$ ,  $X_0^* \oplus X_1 = \gamma$  and  $X_0 \oplus X_1^* = \gamma'$  (as in these cases  $X_1 \oplus X_1^* = (X_0^* \oplus \gamma) \oplus (X_0 \oplus \gamma') = \gamma \oplus \gamma' \oplus \beta$ ) hold with probability  $p(\alpha, \beta, \Delta K) \cdot q(\gamma, \delta, \Delta K) \cdot q(\gamma', \delta, \Delta K)$ . Since the probability of related-key truncated differential  $\gamma \oplus \gamma' \oplus \beta \rightarrow \alpha' (\in D)$  for  $(E^0)^{-1}$  under related-key difference  $\Delta K$  is  $p(D, \gamma \oplus \gamma' \oplus \beta, \Delta K)$ , the probability that satisfies the last  $D$  test in the above process is

$$Pr[BOO-2] = \sum_{\beta, \gamma, \gamma'} p(\alpha, \beta, \Delta K) \cdot q(\gamma, \delta, \Delta K) \cdot q(\gamma', \delta, \Delta K) \cdot p(D, \beta \oplus \gamma \oplus \gamma', \Delta K).$$

Since for a random cipher, the  $D$  test holds with probability  $|D| \cdot 2^{-n}$ ,  $Pr[BOO-2] > |D| \cdot 2^{-n}$  must hold for the related-key boomerang distinguisher to work. Moreover,  $2 \cdot (Pr[BOO-2])^{-1}$  chosen plaintext pairs and  $2 \cdot (Pr[BOO-2])^{-1}$  adaptively chosen ciphertext pairs produce at least 2 quartets that satisfy the  $D$  test.

Related-key rectangle and boomerang distinguishers based on four related keys are the same as the previous distinguishers except for using four related keys  $K, K^*, K'$  and  $K'^*$  such that  $K \oplus K^* = K' \oplus K'^* = \Delta K$  and  $K \oplus K' = K^* \oplus K'^* = \Delta K'$ , i.e., the plaintexts  $P_0, P_0^*, P_1^*$  and  $P_1$  in the previous 2-key rectangle and boomerang processes are encrypted using the keys  $K, K^*, K'$  and  $K'^*$ , respectively. Similarly, we can calculate the probabilities of related-key rectangle and boomerang distinguishers and the required data complexity. For a related-key rectangle distinguisher, the probability is

$$Pr[REC-4] = \sum_{\beta, \beta', \gamma} p(\alpha, \beta, \Delta K) \cdot p(\alpha, \beta', \Delta K) \cdot 2^{-n} \cdot q(\gamma, D, \Delta K') \cdot q(\gamma \oplus \beta \oplus \beta', D, \Delta K').$$

If the number of plaintext pairs  $(P_i, P_i^*)$  (related to  $(K, K^*)$ ) and  $(P'_i, P'^*_i)$  (related to  $(K', K'^*)$ ) we collect is  $m$ , respectively, we can generate  $m^2$  quartets and thus we have at least  $m^2 \cdot Pr[REC-4]$  ciphertext quartets which satisfy the  $D$  test. Therefore, in order to get at least 2 such quartets we need about  $4 \cdot (Pr[REC-4])^{-1/2} \cdot 2^{1/2}$  chosen plaintext queries.

For a related-key boomerang distinguisher, the probability<sup>1</sup> is

$$Pr[BOO-4] = \sum_{\beta, \gamma, \gamma'} p(\alpha, \beta, \Delta K) \cdot q(\gamma, \delta, \Delta K') \cdot q(\gamma', \delta, \Delta K') \cdot p(D, \beta \oplus \gamma \oplus \gamma', \Delta K).$$

So the data requirement to generate at least two good quartets is about  $2 \cdot (Pr[BOO-4])^{-1}$  chosen plaintext pairs and  $2 \cdot (Pr[BOO-4])^{-1}$  adaptively chosen ciphertext pairs.

## 4 Related-Key Rectangle and Boomerang Attacks on Encryption Modes of MD4, MD5 and HAVAL

In this section, we present related-key rectangle and boomerang attacks on the encryption modes of MD4, MD5 and HAVAL. First, we present related-key rectangle and boomerang distinguishers of MD4 and show how to use them to distinguish MD4 from a random cipher. Second, we apply related-key rectangle and boomerang attacks to MD5 and HAVAL.

### 4.1 Cryptanalysis of MD4

In MD4 the message expansion algorithm is a linear function in each pass every message word is used exactly once in a specified order. It means that in the encryption mode of MD4 the key scheduling algorithm is the same linear function of the message expansion algorithm of MD4. We exploit the simple linear key scheduling algorithm in our distinguishers. The main idea behind our constructions of related-key rectangle and boomerang distinguishers based on two related keys is to give a difference in one key word whose interval between the first and third passes is as wide as possible. Let the round numbers involved in such a key word in the three passes be  $r_1, r_2$  and  $r_3$ . Then we can make probability-one differentials for rounds  $r_1 \sim r'_2$  and  $r'_2 \sim r_3$  by giving appropriate differences  $\alpha$  and  $\gamma$ , respectively, where  $r'_2$  is a certain number between  $r_1$  and  $r_2$ . Therefore, in order to find distinguishers with high probabilities we should find one key word for which the interval of  $r_1 \sim r_3$  is as wide as possible.

In our observation giving a difference in the 3-rd key word provides the best probabilities to our distinguishers, which are described as follows. In MD4 there exist a related-key differential characteristic  $(0, e_{31}, 0, 0) \rightarrow (0, 0, 0, 0)$  for rounds  $0 \sim 27$  with probability  $2^{-2}$  (denoted  $p$ ) and a related-key differential characteristic  $(e_{31}, 0, 0, 0) \rightarrow (e_2, e_{5,17,26,28}, e_{13,22}, e_{11})$  for rounds  $28 \sim 47$  with probability  $2^{-7}$  (denoted  $q$ ) under key difference  $\Delta K = (0, 0, 0, \Delta K^3 = e_{31}, 0, \dots, 0)$ , where  $e_i$  represents a 32-bit word that has 0's in all bit positions except for bit  $i$  and  $e_{i_1, \dots, i}$  represents  $e_{i_1} \oplus \dots \oplus e_{i_i}$  (in our notation the right most bit is referred to as the 0-th bit, i.e., the least significant bit). See Table 3 for more details. The

<sup>1</sup> If the set  $D$  has a single element  $\alpha$  in  $Pr[BOO-4]$  and the set  $D$  has a single element  $\delta$  in  $Pr[REC-4]$ , it holds  $Pr[REC-4] = 2^{-n} \cdot Pr[BOO-4]$ . This relationship also holds between  $Pr[BOO-2]$  and  $Pr[REC-2]$ . We use these relationships to estimate  $Pr[REC-2]$  and  $Pr[REC-4]$  in our attacks.

**Table 3.** Related-Key Distinguishers of MD4 (Two Related Keys)

Round ( <i>i</i> )	$\Delta A$	$\Delta B$	$\Delta C$	$\Delta D$	$\Delta K$	Prob.
0	0	$e_{31}$	0	0	0	1
1	0	0	$e_{31}$	0	0	$2^{-1}$
2	0	0	0	$e_{31}$	0	$2^{-1}$
3	$e_{31}$	0	0	0	$e_{31}(= \Delta K^3)$	1
4	0	0	0	0	0	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
27	0	0	0	0	0	1
	0	0	0	0		$p = 2^{-2}$
28	$e_{31}$	0	0	0	$e_{31}(= \Delta K^3)$	1
29	0	0	0	0	0	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
43	0	0	0	0	0	1
44	0	0	0	0	$e_{31}(= \Delta K^3)$	1
45	0	$e_2$	0	0	0	$2^{-1}$
46	0	$e_{11}$	$e_2$	0	0	$2^{-2}$
47	0	$e_{13,22}$	$e_{11}$	$e_2$	0	$2^{-4}$
	$e_2$	$e_{5,17,26,28}$	$e_{13,22}$	$e_{11}$		$q = 2^{-7}$
<i>REC-2</i>	$(0 \rightarrow 27)^2, (28 \rightarrow 45)^2$					$\Pr[\text{REC-2}] \approx 2^{-134}$
<i>BOO-2</i>	$(0 \rightarrow 27), (47 \rightarrow 28)^2, (27 \rightarrow 3)$					$\Pr[\text{BOO-2}] \approx 2^{-16}$
<i>BOO-2</i>	Fixed $K^{0,1,2,7,11,15}, (3 \rightarrow 27), (44 \rightarrow 28)^2, (27 \rightarrow 3)$					$\Pr[\text{BOO-2}] = 1$

notation used in Table 3 is essential in our distinguishing attacks. The *REC-2* and *BOO-2* rows represent probabilities which will be used in related-key rectangle and boomerang attacks, respectively and the *BOO<sup>W</sup>-2* row represents a weak key class as well as a probability which will be used in a related-key boomerang attack under a weak key class. The notation  $(r \rightarrow r')^1$  or  $^2$  means related-key differentials for rounds from  $r$  to  $r'$  (which have the fixed difference in round  $r$  or  $r'$  described in the table) used in our distinguishers. Here, the superscript 1 or 2 represents how many times related-key differentials are used in our distinguishers. Note that if  $r > r'$  then the related-key differential works through decryption process.

In order to estimate  $Pr[\text{BOO-2}]$  we have carried out experiments on a number of related keys with  $2^{23}$  chosen plaintext pairs and  $2^{23}$  adaptively chosen ciphertext pairs each and we have observed 136, 115, 136, 125, 132, 130, 132, 131, 119, 144,  $\dots$  boomerangs returning for each related-key. This simulation result provides that the probability  $Pr[\text{BOO-2}]$  is approximately  $2^{-16}$  (which can be also calculated from the probabilities of related-key differential characteristics in Table 3). We can use the value of  $Pr[\text{BOO-2}]$  or the probabilities of related-key differential characteristics in Table 3 to obtain the probability  $Pr[\text{REC-2}]$ .

We now present a distinguishing attack of the encryption mode of MD4 using a related-key rectangle distinguisher in Table 3. As stated in Table 3, in this attack we use  $Pr[\text{REC-2}] \approx 2^{-134}$ , which is derived from  $p = 2^{-2}$  and  $q' = 2^{-1}$  (the  $q'$  is the probability for rounds  $28 \sim 45$  in Table 3). In order to use  $p = 2^{-2}$  we should collect plaintext pairs  $(P_i, P_i^*)$  which satisfy not only the  $(0, e_{31}, 0, 0)$

difference but also  $c_{31} = d_{31} = 0$ , where  $c_j$  and  $d_j$  represent the  $j$ -th bits of words  $C$  and  $D$  of  $P_i$ , respectively. Moreover, since we use  $q' = 2^{-1}$  for rounds  $28 \sim 45$  in our attack, our desired  $\delta$  after round 47 can be any one of the differences which can be derived from the input difference of round 46,  $(0, e_{11}, e_2, 0)$ . It is easy to see that the number of all possible  $\delta$ 's is at most  $2^{36}$ . We denote the set of all these possible  $\delta$ 's by  $\mathcal{O}$ . Next we describe our distinguishing attack on the encryption mode of MD4 using the related-key rectangle distinguisher.

1. Prepare  $2^{68}$  plaintext pairs  $(P_i, P_i^*)$ ,  $i = 0, 1, \dots, 2^{68} - 1$  with difference  $(0, e_{31}, 0, 0)$  and  $c_{31} = d_{31} = 0$ .
2. With a chosen plaintext attack, obtain the  $2^{68}$  corresponding ciphertext pairs  $(C_i, C_i^*)$ , i.e.,  $C_i = E_K(P_i)$  and  $C_i^* = E_{K^*}(P_i^*)$ , where  $E$  is either MD4 or a randomly chosen cipher and  $K \oplus K^* = (0, 0, 0, \Delta K^3 = e_{31}, 0, \dots, 0)$ .
3. If there exists at least one ciphertext quartet such that  $C_i \oplus C_i^*, C_i^* \oplus C_j \in \mathcal{O}$  for  $0 \leq i \neq j \leq 2^{68} - 1$ , we identify  $E$  as MD4. Otherwise, we identify  $E$  as a randomly chosen cipher.

From the  $2^{68}$  plaintext pairs we obtain  $2^{135}$  quartets. Since our related-key rectangle distinguisher has a probability of  $(2^{-2})^2 \cdot (2^{-1})^2 \cdot 2^{-128} = 2^{-134}$ , if  $E$  is MD4, this attack will succeed with a probability of  $1 - (1 - 2^{-134})^{2^{135}} \approx 0.86$ . On the other hand, in case  $E$  is a randomly chosen cipher, the probability that each ciphertext quartet satisfies one of all possible  $\delta$ 's is less than  $(\frac{2^{36}}{2^{128}})^2 = 2^{-184}$ , so, in this case this attack will succeed with a probability of  $(1 - 2^{-184})^{2^{135}} \approx 1$ . Therefore, the success rate of this attack is about  $\frac{1}{2} \cdot 0.86 + \frac{1}{2} \cdot 1 = 0.93$ .

Based on the foregoing two related-key differentials we can also exploit a boomerang technique to distinguish MD4 from a randomly chosen cipher. In a boomerang technique we use  $Pr[BOO-2] \approx 2^{-16}$ . Since we use related-key differentials for rounds  $27 \sim 3$ , our desired  $\alpha$  before round 0 can be any one of the differences which can be derived from the input difference of round 3,  $(e_{31}, 0, 0, 0)$ , through the inverse direction. It is easy to see that the all possible  $\alpha$ 's are  $(0, e_{31}, 0, 0)$ ,  $(e_{31}, e_{31}, 0, 0)$ ,  $(0, e_{31}, e_{31}, e_{31})$  and  $(e_{31}, e_{31}, e_{31}, e_{31})$ . In order to produce two boomerangs this attack exploits  $2^{17}$  plaintext pairs with desired conditions and  $2^{17}$  adaptively chosen ciphertext pairs. We distinguish MD4 from a random cipher by checking whether or not there exists at least one plaintext pair corresponding to adaptively chosen ciphertext pair that satisfy one of  $\alpha$ 's.

Since our related-key boomerang distinguisher has a probability of  $2^{-2} \cdot (2^{-7})^2 = 2^{-16}$ , if  $E$  is MD4 this attack will succeed with a probability of  $1 - (1 - 2^{-16})^{2^{17}} \approx 0.86$ . In order to verify this estimation we have performed hundreds of simulations using  $2^{18}$  chosen plaintext and adaptively chosen ciphertext pairs each (in each simulation we used randomly chosen related keys and plaintext/ciphertext pairs). In our simulations we could check that about 88 among 100 tests satisfy the above distinguishing attack on average. This result is quite similar to our estimation.

On the other hand, if  $E$  is a randomly chosen cipher, the probability that each plaintext pair satisfies one of the four  $\alpha$ 's is  $\frac{4}{2^{128}} = 2^{-126}$ , so, in this case this



attack will succeed with a probability of  $(1 - 2^{-126})^{2^{17}} \approx 1$ . Therefore, the success rate of this attack is almost same as that of the related-key rectangle attack.

Moreover, we can increase the boomerang probability from  $2^{-16}$  to 1 by using some weak key class. Assume that the first three and the last three round keys  $K^0, K^1, K^2, K^7, K^{11}$  and  $K^{15}$  are fixed and known to the attacker. Then we can use  $p' = 1$  for rounds  $3 \sim 27$  and  $q' = 1$  for rounds  $44 \sim 28$  in our attack under the weak key class assumption. If  $E$  is MD4, the distinguishing attack will succeed with probability one (we have checked with thousands of simulations that this attack always works in MD4), but if  $E$  is a randomly chosen cipher, this attack will succeed with probability  $1 - 2^{-128}$ . Therefore, the success rate of this attack is almost 1. The details of the boomerang attack procedures are given in [11].

Similarly, we can construct related-key rectangle and boomerang distinguishers based on four related keys and distinguish MD4 from a randomly chosen cipher by using them. As a compensation of the use of four related keys, these attacks are more efficient than those with two related keys. See the full version of the paper [11] (Table 5 in Appendix B) for the distinguishers and Table 1 for the results.

## 4.2 Cryptanalysis of MD5 and HAVAL

Similarly, in the MD5 and HAVAL attacks, we first find consecutive two related-key differential characteristics with high probabilities which are independent of each other, and then we can estimate the probability  $Pr[BOO-k]$  on the basis of those differential characteristics by a series of simulations, where  $k$  is 2 or 4. As for 5-pass HAVAL, we can carry out an experiment on a reduced-round variant (which is truncated for the first and the last several rounds) to get  $Pr[BOO-k]$  for the reduced variant and then we can use the obtained value as well as probabilities for the truncated rounds of the consecutive two related-key differential characteristics (which were found in the first stage) to estimate  $Pr[BOO-2]$  for the full 5-pass HAVAL. Once we get the probability  $Pr[BOO-k]$ , we can estimate the probability  $Pr[REC-k]$  by using the relationship between them described in Section 3. See [11] (Appendix C and Appendix D) for the distinguishers of MD5 and HAVAL and Table 1 for the results. We also refer the readers to [11] (Appendix A) for an example of a boomerang quartet for MD5.

## 5 Conclusion

In this paper, we have applied the recently proposed related-key rectangle and boomerang attacks to the encryption modes of MD4, MD5 and HAVAL. The MD4, MD5 and HAVAL used in encryption modes are all vulnerable to those attacks, in particular, they can be broken by related-key boomerang attacks in a real time. The attacks have been experimentally tested and run milliseconds on a PC.

Our results show that one should be very careful when using existing hash functions in encryption mode.

## References

1. E. Biham and R. Chen, *Near-Collisions of SHA-0*, Advances in Cryptology – Proceedings of CRYPTO 2004, LNCS 3152, pp. 290-305, Springer-Verlag, 2004.
2. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet and W. Jalby, *Collisions of SHA-0 and Reduced SHA-1*, Advances in Cryptology – Proceedings of EUROCRYPT 2005, LNCS 3494, pp. 22-35, Springer-Verlag, 2005.
3. E. Biham, O. Dunkelman and N. Keller, *Rectangle Attacks on 49-Round SHACAL-1*, Proceedings of Fast Software Encryption 2003, LNCS 2887, pp. 22-35, Springer-Verlag, 2003.
4. E. Biham, O. Dunkelman and N. Keller, *Related-Key Boomerang and Rectangle Attacks*, Advances in Cryptology – Proceedings of EUROCRYPT 2005, LNCS 3494, pp. 507-525, Springer-Verlag, 2005.
5. H. Dobbertin, A. Bosselaers and B. Preneel, *RIPEMD-160: A Strengthened Version of RIPEMD*, Proceedings of Fast Software Encryption 1996, LNCS 1039, pp. 71-82, Springer-Verlag, 1996.
6. H. Handschuh, L.R. Knudsen and M.J. Robshaw, *Analysis of SHA-1 in Encryption Mode*, Proceedings of CT-RSA 2001, LNCS 2020, pp. 70-83, Springer-Verlag, 2001.
7. H. Handschuh and D. Naccache, *SHACAL : A Family of Block Ciphers*, Submission to the NESSIE project, 2002.
8. J. Kim, D. Moon, W. Lee, S. Hong, S. Lee and S. Jung, *Amplified Boomerang Attack against Reduced-Round SHACAL*, Advances in Cryptology – ASIACRYPT 2002, LNCS 2501, pp. 243-253, Springer-Verlag, 2002.
9. J. Kim, G. Kim, S. Hong, S. Lee and D. Hong, *The Related-Key Rectangle Attack - Application to SHACAL-1*, Proceedings of Australian International Conference on Information Security and Privacy 2004, LNCS 3108, pp. 123-136, Springer-Verlag, 2004.
10. J. Kim, G. Kim, S. Lee, J. Lim and J. Song, *Related-Key Attacks on Reduced Rounds of SHACAL-2*, Proceedings of INDOCRYPT 2004, LNCS 3348, pp. 175-189, Springer-Verlag, 2004.
11. J. Kim, A. Biryukov, B. Preneel and S. Lee, *On the Security of Encryption Modes of MD4, MD5 and HAVAL*, Cryptology ePrint Archive, Report 2005/327, Available Online at <http://eprint.iacr.org/2005/327.ps>.
12. S. Hong, J. Kim, G. Kim, J. Sung, C. Lee and S. Lee, *Impossible Differential Attack on 30-Round SHACAL-2*, Proceedings of INDOCRYPT 2003, LNCS 2904, pp. 97-106, Springer-Verlag, 2003.
13. S. Hong, J. Kim, S. Lee and B. Preneel, *Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192*, Proceedings of Fast Software Encryption 2005, to appear.
14. R.L. Rivest, *The MD4 Message Digest Algorithm*, Advances in Cryptology – Proceedings of CRYPTO 1990, Springer-Verlag, 1991, 303-311.
15. R.L. Rivest, *The MD5 Message Digest Algorithm*, Request for Comments (RFC 1320), Internet Activities Board, Internet Privacy Task Force, 1992.
16. M.J.O. Saarinen, *Cryptanalysis of Block Ciphers Based on SHA-1 and MD5*, Proceedings of Fast Software Encryption 2003, LNCS 2887, pp. 36-44, Springer-Verlag, 2003.

17. Y. Shin, J. Kim, G. Kim, S. Hong and S. Lee, *Differential-Linear Type Attacks on Reduced Rounds of SHACAL-2*, Proceedings of Australian International Conference on Information Security and Privacy 2004, LNCS 3108, pp. 110-122, Springer-Verlag, 2004.
18. X. Wang and H. Yu, *How to Break MD5 and Other Hash Functions*, Advances in Cryptology – Proceedings of EUROCRYPT 2005, LNCS 3494, pp. 19-35, Springer-Verlag, 2005.
19. X. Wang, X. Lai, D. Feng, H. Chen and X. Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*, Advances in Cryptology – Proceedings of EUROCRYPT 2005, LNCS 3494, pp. 1-18, Springer-Verlag, 2005.
20. X. Wang, H. Yu and Y.L. Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology – Proceedings of CRYPTO 2005, LNCS 3621, pp. 1-16, Springer-Verlag, 2005.
21. X. Wang, Y.L. Yin and H. Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology – Proceedings of CRYPTO 2005, LNCS 3621, pp. 17-36, Springer-Verlag, 2005.
22. H. Yoshida, A. Biryukov, C. De Cannière, J. Lano and B. Preneel, *Non-randomness of the Full 4 and 5-pass HAVAL*, Proceedings of SCN 2004, LNCS 3352, pp. 324-336, Springer-Verlag, 2005.
23. Y. Zheng, J. Pieprzyk and J. Seberry, *HAVAL-A One-way Hashing Algorithm with Variable Length of Output*, Advances in Cryptology – Proceedings of AUSCRYPT 1992, LNCS 718, pp. 83-104, Springer-Verlag, 1993.
24. RIPE, Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evaluation(RIPE-RACE 1040), LNCS 1007, 1995.
25. U.S. Department of Commerce. *FIPS 180-1: Secure Hash Standard*, Federal Information Processing Standards Publication, N.I.S.T., April 1995.
26. U.S. Department of Commerce. *FIPS 180-2: Secure Hash Standard*, Federal Information Processing Standards Publication, N.I.S.T., August 2002.

# Cryptanalysis of PASS II and MiniPass

Bok-Min Goi<sup>1</sup>, Jintai Ding<sup>2</sup>, and M.U. Siddiqi<sup>1,\*</sup>

<sup>1</sup> Centre for Cryptography and Information Security (CCIS),  
Faculty of Engineering, Multimedia University,  
63100 Cyberjaya, Malaysia

`bngoi@mmu.edu.my`

<sup>2</sup> Department of Mathematical Sciences,  
University of Cincinnati, Cincinnati,  
OH 45221-0025 USA

`ding@math.uc.edu`

**Abstract.** In ACISP '00, Wu *et al.* proposed attacks to break the Polynomial Authentication and Signature Scheme (PASS), in particular, they are able to generate valid authentication transcripts and digital signatures without knowing the private key and any previous transcripts/signatures. They showed that PASS can be broken with around  $2^{38.3}$  trials. In this paper, we analyze the security of the improved versions of PASS; viz. PASS II and MiniPASS, and extend the Wu *et al.*'s attacks to PASS II and MiniPASS to break them. Furthermore, we discuss why and how these schemes are broken from the view point of the structure of cryptosystems and point out the fundamental weakness behind.

**Keywords:** Authentication scheme, digital signature scheme, cryptanalysis, NTRU, partial polynomial evaluation.

## 1 Introduction

For electronic communications and commerce in a common networked and open environment, security issues are always the main concern. In this context, public key authentication and digital signature schemes that provide authentication and non-repudiation services to communicating parties have been of steadily increasing interest in cryptographic studies. They are not only need to be secure, but also have to be fast and can be implemented on low power computing devices, i.e. low-cost smart cards and RFID devices. Since year 1996, researchers from NTRU Cryptosystem Inc. have proposed a group of fast public key cryptosystem based on the hard problems of partial evaluation of constrained polynomial over polynomial rings. These comprise of NTRU public key encryption algorithm [3], NTRUSign digital signature scheme [1], Polynomial Authentication and Signature Scheme, PASS [2], and its variant PASS II [5] and MiniPASS [4]. The hard problem underlying this group of cryptosystem can be related to short vectors in a lattice due to properties of short polynomials used in the system.

---

\* The first author acknowledges the Malaysia IRPA grant (04-99-01-00003-EAR).

In ACISP '00, Wu *et al.* presented two attacks on PASS [6]. In particular, they are able to generate valid authentication transcripts and digital signatures without knowing the private key or previous transcripts / signatures. Though their first attack can be easily prevented with some proper parameter settings, PASS can still be broken with around  $2^{38.3}$  trials under their second attack.

In this paper, we further analyze the security of the improved versions of PASS; viz. PASS II and MiniPASS, and extend the Wu *et al.*'s attacks to PASS II and MiniPASS, and show how to break them efficiently as well. Furthermore, we discuss how and why these schemes are broken from the point view of the structure of cryptosystems and point out the fundamental weakness behind which allows these attacks, namely the participation of a verifier in the process setting up the challenge. Therefore, though we believe that the concept behind the construction of PASS, namely the hard problems of partial evaluation of constrained polynomials over polynomial rings, is still correct and sound, any new system based on the same idea as that of PASS needs to overcome such a fundamental weakness in order to work securely, in particular, in terms of resisting the type of attack like that of the Wu *et al.*'s attacks.

The paper is organized as following. In the next section, we will outline the authentications systems PASS, PASS II and MiniPASS [2,5,4]. We then briefly introduce the basic idea of the Wu *et al.*'s attacks [6] in Section 3. In Section 4, we will present the details of our attack to break the PASS II and MiniPASS, which is an extension of the idea of Wu *et al.*. Then, we elaborate the structure analysis of PASS cryptosystems in Section 5. Finally, we conclude in Section 6.

## 2 An Overview of PASS and PASS II

### 2.1 Preliminary and Notations

The ring of truncated polynomials is defined as:  $R = (Z/qZ)[X]/(X^N - 1)$ , where  $q$  and  $N$  are co-relative prime integer. Note that all arithmetic operations are in  $R$  in this paper. The resultant coefficients are reduced modulo  $q$  and exponents are reduced modulo  $N$ . For the proposed schemes in [2, 5, 4],  $q$  and  $N$  were chosen be a prime number (i.e., 769) and a divisor of  $(q - 1)$  (i.e., 768), respectively. A element  $g \in R$  is denoted as a polynomial with degree of  $(N - 1)$  and its coefficients  $g_i \in Z/qZ$ , for  $i = 0, 1, \dots, (N - 1)$ , as  $g(X) = \sum_{i=0}^{N-1} g_i X^i = g_0 + g_1 X + g_2 X^2 + \dots + g_{N-1} X^{N-1}$ .

For each element  $\alpha \in Z/qZ$ ,  $g(\alpha)$  means that substituting the variable  $X$  in the polynomial  $g$  with the value  $\alpha$  and the result is reduced modulo  $q$ . For multiplication of two polynomials in  $R$ , since the exponents of the product are reduced modulo  $N$ , thus it is a cyclic convolution multiplication. For example, given  $f, g \in R$ , the product  $h = fg$  in  $R$  will be  $h(X) = \sum_{i=0}^{N-1} f_i X^i \sum_{j=0}^{N-1} g_j X^j = \sum_{k=0}^{N-1} h_k X^k$ , where  $h_k = \sum_{i+j=k \bmod N} f_i g_j$ . Note that if  $\alpha^N = 1 \pmod q$ , then  $h(\alpha) = f(\alpha)g(\alpha)$ . Informally, a short polynomial  $g$  is a polynomial with small norm value  $\|g\|_2 = \sqrt{\sum_{i=0}^{N-1} g_i^2}$ . For example, it may contain many zero coefficients and few coefficients with small value (i.e., -1 or +1). A polynomial  $g$  in

**Table 1.** The notations

$P$	The prover
$V$	The verifier
$c_o$	The 80-bit challenge
$M$	The message to be signed
$S$	The set of $t$ distinct non-zero elements of $\alpha \in Z/qZ$ , where $\alpha^N = 1 \pmod q$ and $\alpha^{-1} \in S$ . Note that $t \approx \frac{q}{2}$
$f_1, f_2$	The private key and its corresponding public key is $(f_1(\alpha), f_2(\alpha))$ for $\alpha \in S$ . Note that only one private polynomial $f$ for PASS II
$g_1, g_2$	The commitment polynomials with corresponding commitment values $(g_1(\alpha), g_2(\alpha))$ for $\alpha \in S$
$d_i$	The polynomial $i$ which contains $d_i$ coefficients equal to each of 1 and $-1$ , and the rest equal to 0, for $i = f, g$ and $c$ . For example, the suggested parameters for PASS are $d_f = d_g = 256, d_c = 1$
$L_i$	The public special subset of $R$ , for $i = f, g, c$ and $h$ . Note that $h$ is computed based on the private key, commitment polynomials and the outputs of <i>Hash</i> . In more detail, $L_f, L_g$ and $L_c$ are those special subsets whose element polynomials contain $d_f, d_g$ and $d_c$ parameters, respectively; whereas, $L_h = h \in R : \ h\ _2 \leq \gamma q$
$Hash(\cdot)$	The special hash function which hashes $c_o$ or $M$ with the commitments to produce some polynomials in $L_c$

$R$  is called *moderately* short if its norm is smaller than a constant times  $q$ , such that  $\|g\|_2 \leq \gamma q$ , where the value of  $\gamma$  is determined by the particular application via experiment.

For ease of explanation, we use the notations similar to those in [2, 6, 5], as shown in Table 1.

## 2.2 PASS

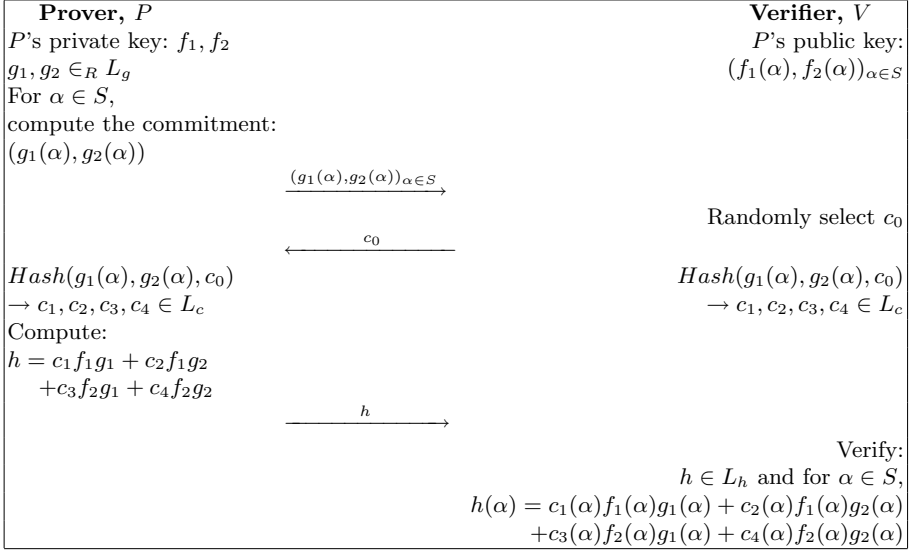
The prover  $P$  randomly chooses two polynomial  $(f_1, f_2)$  from  $L_f$  as his private key. Then he generates his public key as  $(f_1(\alpha), f_2(\alpha))$  for  $\alpha \in S$ . The private key is well protected due to the fact that recovering a short polynomial in  $R$  by providing only a certain subset values of the polynomial is a hard problem. Furthermore, it is also difficult to find another two short polynomials  $(f'_1, f'_2)$ , which for  $\alpha \in S$ , must satisfy the conditions:

$$f'_1(\alpha) = f_1(\alpha) \quad \text{and} \quad f'_2(\alpha) = f_2(\alpha). \quad (1)$$

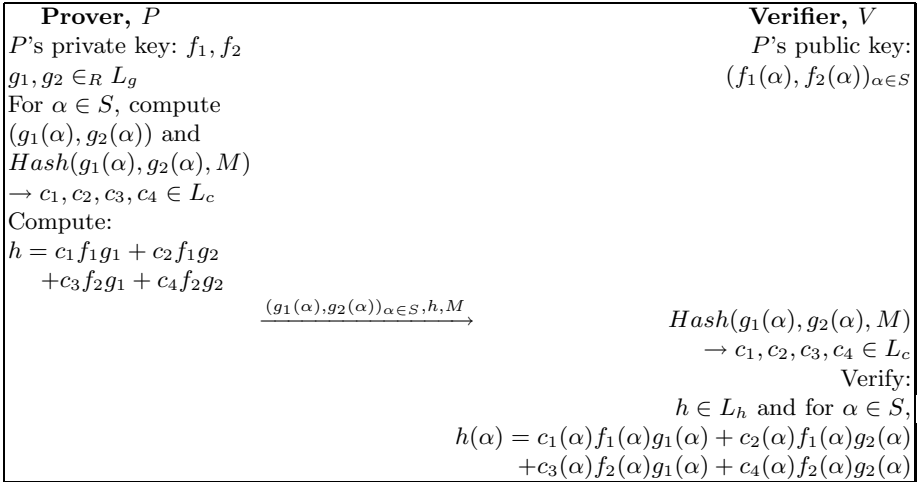
We refer readers to [2] for more security analysis of PASS. For simplicity, we depict the PASS authentication scheme and the PASS digital signature scheme in Fig. 1 and Fig. 2, respectively.

## 2.3 PASS II and MiniPASS

In [5], Hoffstein and Silverman have proposed another improved version of PASS, called as PASS II. PASS II only involves single polynomial  $f$  in key pair creation



**Fig. 1.** PASS authentication scheme



**Fig. 2.** PASS signature scheme

phase and only one set of values of  $(g_1(\alpha))_{\alpha \in S}$  required for the verifier  $V$ . Therefore, it can further reduce the computational complexity and communication requirements. In CHES '00 [4], based on PASS II, the same authors presented a scheme which is called as MiniPASS and described how it can be implemented in highly constrained devices.

<p><b>Prover, <math>P</math></b>  <math>P</math>'s private key: <math>f</math>  <math>g_1, g_2 \in_R L_g</math>                  For <math>\alpha \in S</math>, compute:  <math>g_1(\alpha)</math> and then,  <math>Hash(g_1(\alpha), M)</math>  <math>\rightarrow c_1, c_2 \in L_c</math>,                  where <math>c_1(\alpha) \neq 0</math>,                  for <math>2 \leq \alpha \leq q - 2</math> and <math>\alpha \notin S</math>.                  Compute:  <math>h = (f + c_1g_1 + c_2g_2)g_2</math></p>	$\xrightarrow{(g_1(\alpha))_{\alpha \in S, h, M}}$	<p><b>Verifier, <math>V</math></b>  <math>P</math>'s public key:  <math>(f(\alpha))_{\alpha \in S}</math></p> <p><math>Hash(g_1(\alpha), M)</math>  <math>\rightarrow c_1, c_2 \in L_c</math>                  where <math>c_1(\alpha) \neq 0</math>                  for <math>2 \leq \alpha \leq q - 2</math> and <math>\alpha \notin S</math>                  Verify:  <math>h \in L_h</math> and for <math>\alpha \in S</math>,  <math>(f(\alpha) + c_1(\alpha)g_1(\alpha))^2 + 4c_2(\alpha)h(\alpha)</math>                  quadratic residue mod <math>q</math></p>
---	--	---

**Fig. 3.** PASS II signature scheme

**Table 2.** Comparison of PASS and PASS II signature scheme

Description	Original PASS [2]	PASS II [5]
Creating key pair	Two private polynomials $f_1$ and $f_2$ with the corresponding public values $(f_1(\alpha), f_2(\alpha))_{\alpha \in S}$ .	Only one private polynomials $f$ with the corresponding public values $(f(\alpha))_{\alpha \in S}$ .
Generating commitment polynomials	Two commitment polynomials $(g_1, g_2)$ . Both $(g_1(\alpha), g_2(\alpha))$ have to be sent to the verifier.	Two commitment polynomials $(g_1, g_2)$ . But only $g_1(\alpha)$ is required to be sent to the verifier.
Computing $h$	$h = c_1f_1g_1 + c_2f_1g_2 + c_3f_2g_1 + c_4f_2g_2$	$h = (f + c_1g_1 + c_2g_2)g_2$
Verification process	Check $h \in L_h$ and the values of $h(\alpha)$ for $\alpha \in S$	Check $h \in L_h$ and $(f(\alpha) + c_1(\alpha)g_1(\alpha))^2 + 4c_2(\alpha)h(\alpha) =$ quadratic residue mod $q$ , for $\alpha \in S$ .
Parameter settings	$d_f = d_g \approx \frac{q}{3}, d_c = 1, \gamma = 2.2$	$d_f \approx \frac{q}{3}, d_g \approx \frac{q}{6}, d_c = 2, \gamma = 1.8$ .
Security level†	With $q = 769$ , PASS is more secure than RSA 1024; With $q = 1153$ , PASS is more secure than RSA 2048.	With $q = 769$ , PASS II is more secure than RSA 512; With $q = 929$ , PASS is more secure than RSA 1024.

† : as claimed by the original inventors in [2] and [5].

The PASS II signature scheme is described in Fig. 3. We omit the PASS II authentication scheme because it can be constructed in a similar way as PASS authentication scheme.



We conclude this section by providing the comparison between the original PASS and PASS II schemes in Table 2. (Note that this comparison could be applied directly to the authentication schemes as well.)

### 3 Two Attacks Due to Wu *et al.*

In [6], Wu *et al.* proposed two attacks on the PASS system. For ease of explanation and also lack of better names, we simply denote the first attack proposed in Section 3 of [6] as Attack 1 and the second attack proposed in Section 4 of [6] as Attack 2, respectively.

#### 3.1 Wu *et al.*'s Attack 1

Wu *et al.* discovered that in order to break PASS which amounts to satisfying the expression:

$$h(\alpha) = c_1(\alpha)f_1(\alpha)g_1(\alpha) + c_2(\alpha)f_1(\alpha)g_2(\alpha) + c_3(\alpha)f_2(\alpha)g_1(\alpha) + c_4(\alpha)f_2(\alpha)g_2(\alpha)$$

for  $\alpha \in S$  in the verification process, is not required to find short polynomials  $(f'_1, f'_2)$  satisfying the condition in Eq. (1). However, they claimed that as long as  $g_1(\alpha) = g_2(\alpha) = 0$  for certain  $\alpha \in S$ , then  $f'_1(\alpha)$  and  $f'_2(\alpha)$  can be of any values for other values of  $\alpha$ . They proved that there are at most  $p$  non-zero elements in  $Z/Z_q$  satisfying  $g(\alpha) \neq 0$ . Note that  $p$  is a divisor of  $N$  and the coefficients of polynomial  $g$  from  $\{-1, 0, +1\}$  are with period  $p$ . Based on this, Wu *et al.* came out with Attack 1 where an attacker can forge the authentication transcript/signature independent of the sizes of  $N$  and  $t$ . An attacker, with such small amount of computation, can fool  $B$  into thinking he is a legitimate counterpart  $A$ , even without  $A$ 's private key and previous communicated transcripts.

**Observations on Attack 1.** Here, we correct a typo in the expression in the proof of Theorem 1 [6], where a polynomial  $g$  with period  $p$  (obviously,  $p$  must be one of the factor of  $N$ ) should be denoted as:

$$g(X) = (g_0 + g_1X + g_2X^2 + \dots + g_{p-1}X^{p-1})(1 + X^p + X^{2p} + \dots + X^{N-p}),$$

but not as:

$$g(X) = (g_0 + g_1X + g_2X^2 + \dots + g_{p-1}X^{p-1})(1 + X^p + X^{2p} + \dots + X^{N/p}).$$

We further remark that, in [6], in order to countermeasure Attack 1, Wu *et al.* suggested  $N$  to be chosen as a prime or with no small factor. However, PASS only works in the special case where  $\alpha^N = 1 \pmod q$  in ring  $R$ . This is to ensure homomorphism mapping. More precisely,  $N$  must be a divisor of  $(q - 1)$  and according to the fact of Fermat's Little Theorem [2], for all non-zero element, we can well define the homomorphic mapping from  $R$  to  $Z/qZ$  as:  $g(X) \rightarrow g(\alpha^{(q-1)/N})$ . Hence,  $N$  has to be determined carefully.

### 3.2 Wu *et al.*'s Attack 2

Wu *et al.* proposed Attack 2 by exploiting the fact that the space of  $L_c$ ,  $|L_c|$  is small (as  $d_c = 1$ ). Attack 2 on the PASS signature scheme is described as:

1. Randomly select  $r_1, r_2 \in R$  such that  $r_1c, r_2c \in L_c$  for  $c \in L_c$ .
2. For  $\alpha \in S$ , compute the two sets  $(\beta_1, \beta_2)$  with no zero element, such that  $\beta_1 = f_1(\alpha) + r_1(\alpha)f_2(\alpha)$  and  $\beta_2 = f_1(\alpha) + r_2(\alpha)f_2(\alpha)$ . Note that  $\beta_i = \{\beta_{i1}, \beta_{i2}, \dots, \beta_{it}\}$  where  $i = \{1, 2\}$ .
3. Compute two polynomial  $(f'_1, f'_2)$  such that  $f'_1(\alpha) = \beta_1$  and  $f'_2(\alpha) = \beta_2$ .
4. Arbitrarily choose  $h_1, h_2 \in R$  and compute the polynomials  $(g_1, g_2)$  which satisfy  $h_1(\alpha) = f'_1(\alpha)g_1(\alpha)$  and  $h_2(\alpha) = f'_2(\alpha)g_2(\alpha)$  for  $\alpha \in S$ . Note that  $(g_1, g_2)$  are not required to be short polynomial and fulfill  $d_g$  requirement. Wu *et al.* showed that these two polynomials can be computed easily in Theorem 3 of [6].
5.  $Hash(g_1(\alpha), g_2(\alpha), M)$  and obtain  $c_1, c_2, c_3, c_4 \in L_c$ .
6. Check whether  $c_3 = r_1c_1$  and  $c_4 = r_2c_2$ . If yes, set  $h' = c_1h_1 + c_2h_2$  and obtain a valid signature of message  $M$ . Otherwise, repeat the attack by going back to Step 3.

Obviously, the obtained  $h'$  is equivalent to  $h$ . Attack 2 will succeed with probability  $\frac{1}{|L_c|^2}$ . More precisely, with the proposed parameters in [2]:  $N = 768$  and  $d_c = 1$ , the PASS can only achieve the security level of 38.3 bits. Therefore, the PASS is not secure.

## 4 Our Attacks on PASS II and MiniPASS

In this section, we extend the Wu *et al.*'s attacks on PASS II and MiniPASS. We prove that they face the same problem as PASS. Namely, both with the proposed parameters cannot achieve 80-bit standard security requirement. Hence, PASS II and MiniPASS are insecure and not sound.

### 4.1 Extended Attack 1

Our first extended attack on PASS II works as follows:

1. Calculate the desired polynomials  $g_2$  by setting appropriate period  $p$ .
2. Determine  $S_1 \subset S$  such that for  $\alpha \in S_1$ ,  $g_2(\alpha) = 0$ . Then, fix  $S_2 = S - S_1$ , such that for  $\alpha \in S_2$ ,  $g_2(\alpha) \neq 0$ .
3. Compute the polynomial  $f'$ , such that  $f'(\alpha) = f(\alpha)$  for  $\alpha \in S_2$ . For  $\alpha \in S_1$ , due to Step 2,  $h(\alpha)$  is always equal to zero regardless of the values of  $f'(\alpha)$ .
4. With the three short obtained polynomials  $(g_1, g_2, f')$ , where  $g_1$  can be set arbitrarily, a valid forged authentication transcript satisfying  $h \in L_h$  and for  $\alpha \in S$ ,  $(f(\alpha) + c_1(\alpha)g_1(\alpha))^2 + 4c_2(\alpha)h(\alpha) = \text{quadratic residue (mod } q)$ , can be produced.

The Step 3 above, in determining the polynomial  $f'$  such that  $f'(\alpha) = f(\alpha)$  for  $\alpha \in S_2$ , is the most costly one. On average, the computation complexity is about  $q^{|S_2|}$ , where  $|S_2|$  is the space of  $S_2$ . However, due to the careful selection of  $g_2$  (e.g., with the period  $p = 6$ ),  $|S_2|$  is quite small (e.g.,  $|S_2| \approx \frac{p}{2}$ ), thus computing  $f'$  turn to be an easy task. In particular, for  $|S_2| = 3$ , even with brute-force to randomly search the coefficients of  $f'$ , we only require around  $q^{|S_2|} = 769^3 = 2^{28.8}$  trials to forge a valid authentication transcripts/signature in PASS II with the proposed parameters,  $q = 769$  in [5]. Therefore, PASS II can only provide 28.8-bit security level under this attack. Note that PASS II is much more vulnerable under our extended attack as compared to Attack 1 on PASS, because only one private polynomial  $f'$  needs to be determined.

### 4.2 Extended Attack 2

In [5], Hoffstein and Silverman have increased the space for  $L_c$  by setting  $d_c = 2$ . (In the original PASS,  $d_c = 1$ .) This is mainly due to only two challenge polynomials  $(c_1, c_2)$  are involved in this improved scheme, but not to counter against Attack 2. They claimed that with the proposed parameters,  $N = 768$ , then  $|L_c| = 2^{36}$ . Hence, the space of challenge should be the space of pairs  $(c_1, c_2)$  of elements of  $|L_c|^2$  and equal to  $2^{72}$ . However, in this subsection, we show that it is not true.

In the PASS II scheme, the authentication transcript/signature is expressed as:

$$\begin{aligned} h &= (f + c_1g_1 + c_2g_2)g_2 \\ &= fg_2 + (c_1g_1 + c_2g_2)g_2 \\ &= fg_2 + c_1(g_1 + rg_2)g_2 \\ &= h_1 + c_1h_2 \end{aligned}$$

where  $c_2 = rc_1$ ,  $h_1 = fg_2$  and  $h_2 = (g_1 + rg_2)g_2$ .

Our second extended attack on the PASS II signature scheme works as follows:

1. Select  $r \in_R R$  such that  $rc \in L_c$  for  $c \in L_c$  (for simplicity, set  $r = 1$ ).
2. Arbitrarily choose  $h_2, g_2 \in_R R$ . Then, for  $\alpha \in S$ , compute the polynomial  $g_1$  satisfying  $h_2(\alpha) = (g_1(\alpha) + r(\alpha)g_2(\alpha))g_2(\alpha)$ . Note that  $g_1$  is not required to be short and fulfill  $d_g$  requirement .
3.  $Hash(g_1(\alpha), M)$  and obtain  $c_1, c_2 \in L_c$ .
4. Check whether  $c_2 = rc_1$ . If yes, compute the polynomials  $h_1$ , which is not required to be short, satisfying  $h_1(\alpha) = f(\alpha)g_2(\alpha)$  and set  $h = h_1 + c_1h_2$ , then generate a valid signature of message  $M$  which is  $(g_1(\alpha), h)_{\alpha \in S}$ . Otherwise, repeat the attack by going back to Step 2.

Surprisingly, we found out that our proposed extended attack on PASS II works even more efficient than the original Attack 2 on PASS, in terms of number of steps and computational complexity. In particular, there is no need to compute

the two sets  $(\beta_1, \beta_2)$  and the two polynomial  $(f'_1, f'_2)$  as in the case of PASS, but only compute one polynomial  $g_1$  that satisfies  $h_2(\alpha) = (g_1(\alpha) + r(\alpha)g_2(\alpha))g_2(\alpha)$ . Furthermore, Attack 2 has to be improved in order to work on PASS for the case where the two sets  $(\beta_1, \beta_2)$  contain zero element. This will affect the success rate of this attack.

However, in our attack, the attacker just need to check whether  $c_2 = rc_1$  for every single trial. The success rate will be  $\frac{1}{|L_c|}$ , but not  $\frac{1}{|L_c|^2}$  as claimed in [5]. More precisely, with the suggested parameters in [5], viz.  $N = 768$  and  $d_c = 2$ , the space of  $L_c$ ,  $|L_c| = \frac{N!}{(N-2d_c)!(d_c!)^2} = 2^{36.3}$ . Therefore, PASS II can only offer the security level of 36-bit, and is insecure and not sound.

SUMMARY: Even with proper parameter settings – on choosing good  $N$  – the PASS and PASS II schemes are still vulnerable under Attack 2. This is mainly due to the *decomposition property* of  $h$ .

## 5 Structure Analysis

One of the original intention of our work is to find ways to improve the PASS cryptosystem and try to make it work, because we think that the basic concept behind the construction of the PASS, which is based on the hard problems of partial evaluation of constrained polynomials over polynomial rings, is still sound. To do so, we start from further understanding why and how the PASS systems are broken from the viewpoint of the structure of cryptosystems.

The key point of the Attack 2 is that it successfully transforms the difficult problem of finding the private keys  $(f_1, f_2)$  given  $f_1(\alpha)$  and  $f_2(\alpha)$  for  $\alpha \in S$  to another easier task – finding  $(g_1, g_2)$  – which does not anymore have to be short polynomials (binary or trinary polynomials with many zero coefficients) and fulfill certain requirements on the degree. For example, as stated in [4], by using Discrete Fourier Transform (DFT) method, every coefficients of a polynomial  $G$  can be determined providing all the values of  $G$  via the well-known formula of DFT. In this case, the Attack 2 shows the system's security is not really based on the hard problems of partial evaluation of constrained polynomials over polynomial rings, but rather something else. If one looks deeper, one should realize that the story does not just stop here. The real reason behind is that  $g_1$  and  $g_2$  are actually provided by the prover, which therefore allows the attacker to choose the  $g_1$  and  $g_2$ . Namely in a PASS authentication system, **a prover actually participates in the process in setting up the challenge**. This is very much like leaking partially the secret automatically to the attacker. Such a weakness is the real cause why Attack 2 worked from the structure point of view. Therefore, this, we believe, is a fundamental structure flaw from the point view of designing a secure cryptosystem, because it gives an attacker automatically an edge in attacking the system from the very beginning. This teaches us a good lesson in designing authentication schemes, namely one should not allow a prover to participate in setting up the challenge.

## 6 Concluding Remarks

In this paper, we have comprehensively analyzed the security of PASS II and MiniPASS. We have proposed two extended attacks due to Wu *et al.* [6] and showed how they work on PASS II and MiniPASS in detail. We have further pointed out the main reason for PASS and its variant are broken is due to some flaws in the structure design of the cryptosystems. However, the concept behind the construction of the PASS, based on the hard problems of partial evaluation of constrained polynomials over polynomial rings, is correct and secure.

At the very beginning, we had some suggestions to improve the system. For example: (1) to ensure that the verifier always knows the prover's commitments prior to receiving any new signature, so that the attacker is unable to control the polynomials  $(g_1, g_2)$  and perform Attack 2; (2) to modify the expression of  $h$  so that it won't face the decomposition problems. However, we realize these suggestions do not fundamentally eliminate the flaw of the system. As of now, we have not been able to come up with any good solution, which, we believe, is a very interesting and challenging question.

## Acknowledgement

We would like to thank anonymous referees for their constructive and detailed comments that have greatly improved this paper.

## References

1. J. Hoffstein, N. Graham, J. Pipher, J. Silverman and W. Whyte. NTRUSign: Digital Signatures Using the NTRU Lattice. In *Proceeding of CT-RSA '03*, LNCS, vol. 2612, Springer-Verlag, pp.122-140, 2003.
2. J. Hoffstein, D. Lieman, J. Silverman. Polynomial Rings and Efficient Public Key Authentication. In *Proceeding of CrypTEC '99*, City University of Hong Kong Press, pp. 7-19, 1999.
3. J. Hoffstein, J. Pipher and J. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In *Proceeding of ANTS III*, LNCS, vol. 1423, Springer-Verlag, pp. 267-288, 1998.
4. J. Hoffstein and J. Silverman. MiniPASS: Authentication and Digital Signatures in a Constrained Environment. In *Proceeding of CHES '00*, LNCS, vol. 1965, Springer-Verlag, pp. 328-339, 2000.
5. J. Hoffstein and J. Silverman. Polynomial Rings and Efficient Public Key Authentication II. Available at [www.ntru.com](http://www.ntru.com).
6. Hongjun Wu, Feng Bao, Dingfeng Ye, Robert Deng. Cryptanalysis of Polynomial Authentication and Signature Scheme. In *Proceeding of ACISP '00*, LNCS, vol. 1841. Springer-Verlag, pp. 278-288, 2000.

# Simple Power Analysis on Fast Modular Reduction with NIST Recommended Elliptic Curves

Yasuyuki Sakai<sup>1</sup> and Kouichi Sakurai<sup>2</sup>

<sup>1</sup> Mitsubishi Electric Corporation,  
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan  
ysakai@iss.isl.melco.co.jp

<sup>2</sup> Kyushu University,  
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan  
sakurai@csce.kyushu-u.ac.jp

**Abstract.** We discuss side channel leakage from modular reduction for NIST recommended domain parameters. FIPS 186-2 has 5 recommended prime fields. These primes have a special form which is referred to as generalized Mersenne prime. These special form primes facilitate especially efficient implementation. A typical implementation of efficient modular reduction with such primes includes extra reduction. The extra reduction in modular reduction can constitute an information channel on the secret exponent. Several researchers have produced unified code for elliptic point addition and doubling in order to avoid a simple power analysis (SPA). However, Walter showed that SPA still be possible if Montgomery multiplication with extra reduction is implemented within the unified code. In this paper we show SPA on the modular reduction with NIST recommended primes, combining with the unified code for elliptic point operations. As Walter stated, our results also indicate that even if the unified codes are implemented for elliptic point operations, underlying field operations should be implemented in constant time. The unified approach in itself cannot be a countermeasure for side channel attacks.

**Keywords:** Side channel analysis, elliptic curve cryptosystem, modular reduction, generalized Mersenne prime, unified code.

## 1 Introduction

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 [9]. Side channel attacks can be often much more powerful than mathematical cryptanalysis. Thus, many papers on side channel cryptanalysis have been published.

### 1.1 Modular Reduction in RSA vs. ECC

In RSA, the modulus  $n$  is a product of two randomly generated primes  $p$  and  $q$ . So the modulus  $n$  can not have a special form which permits fast implementation. Montgomery modular multiplication [12], that works with any odd modulus, is a most popular method to achieve fast implementation in the RSA settings. Montgomery reduction includes, at the final step of the computation, an “*extra reduction*” which depends on input value. If the intermediary result of the multiplication is greater than the modulus  $n$ , an extra reduction has to be performed. The extra reduction leads to differences between running times needed for various input values and gives side channel leakage [9,1,7,13,16,17].

On the other hand, in the elliptic curve cryptosystem (ECC) over a prime field  $\mathbb{F}_p$ , the modulus is a domain parameter  $p$ . Before we implement ECC, we must select elliptic curve domain parameters so as to secure against mathematical cryptanalysis. Moreover we must also select domain parameters which permit efficient implementation. These selection can be influenced by security consideration, application platform, processing speed, memory constraint and gate count. For ECC over  $\mathbb{F}_p$ , it is a popular technique to select a special form of prime  $p$  for the field order. The lower level of arithmetics in ECC over  $\mathbb{F}_p$  are modular arithmetics of modulo  $p$ . Solinas gave fast modular reduction methods for generalized Mersenne prime [15]. NIST provides 5 recommended prime fields in FIPS 186-2 [2]. All the recommended elliptic curve domain parameters over  $\mathbb{F}_p$  select generalized Mersenne primes. Montgomery multiplication can also be applied to ECC over  $\mathbb{F}_p$ .

### 1.2 SPA on Unified Code with Modular Reduction: Walter’s Result

The most commonly used algorithm for computing elliptic point multiplication is the binary method. Suppose that the doubling of a point and the addition of two different points are implemented with different formulae, these two operations may then be distinguished by simple power analysis (SPA). A unified approach [3,6,8,10], which unify the standard formulae for point addition and point doubling, offers generic solutions for preventing SPA. Brier and Joye proposed a “unified code”, that unifies the standard formulae for point addition and point doubling on Weierstrass form of an elliptic curve [3]. However, Walter pointed out that SPA may still be possible on the unified code combining with Montgomery multiplication [12] if an extra reduction is included [18]. Walter’s results indicate that one should use a constant time implementation of modular multiplication, whatever multiplier is used [18].

### 1.3 Our Contribution

Walter’s analysis was on Montgomery multiplication for arithmetic in  $\mathbb{F}_p$ . However, as we have stated before, in implementation of ECC over  $\mathbb{F}_p$ , we can apply a dedicated algorithm of modular reduction for a special form of modulus such as the generalized Mersenne prime. A dedicated modular reduction provides faster

implementation than Montgomery multiplication. As we will see in the later section, algorithms of modular reduction for generalized Mersenne prime may also lead an extra reduction depending on the input value.

In this paper, we discuss SPA against ECC using unified code combining with dedicated modular reduction algorithms for generalized Mersenne primes. NIST recommended elliptic domain parameters [2] are examined. We will show that such an implementation is vulnerable against SPA if the extra reduction is included. Assuming the extra reduction can be distinguished by monitoring power consumption during the elliptic point multiplication, the elliptic point operations, point addition and doubling, should be revealed to the attacker. We will also show that  $p_{192} = 2^{192} - 2^{64} - 1$  and  $p_{384} = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$ , which is a field order of NIST recommended domain parameters, can particularly be susceptible to SPA.

## 2 Modular Reduction

The exponentiation is the most expensive part in implementation of public key cryptosystems. In RSA cryptosystems and ECC over a prime field  $\mathbb{F}_p$ , modular multiplication is the dominant operation in the exponentiation. Therefore, it is very attractive to provide algorithms that allow efficient implementation of modular multiplication. Montgomery's method [12] performs modular multiplication without a division instruction which is an expensive operation in almost processors. Thus Montgomery multiplication can achieve computational speed-up and is often used in RSA cryptosystems. In the RSA settings, for security reason, the modulus  $n$  is a product of two randomly generated private primes  $p$  and  $q$ . Thus the modulus  $n$  can not have a special form.

On the other hand, in ECC over  $\mathbb{F}_p$ , the modulus is a prime number  $p$  which is a domain parameter. It is not required to select randomly generated  $p$ . It is preferred to generate  $p$  of special form to achieve fast implementation. In this section we give a brief description of fast modular reduction with special modulus such as the generalized Mersenne prime.

### 2.1 Generalized Mersenne Prime

In FIPS 186-2 NIST provides 5 recommended prime fields [2]. The order of the fields are shown below.

$$\begin{aligned} \text{P-192: } p_{192} &= 2^{192} - 2^{64} - 1 \\ \text{P-224: } p_{224} &= 2^{224} - 2^{96} + 1 \\ \text{P-256: } p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\ \text{P-384: } p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\ \text{P-521: } p_{521} &= 2^{521} - 1 \end{aligned}$$

These recommended primes have a special form, which are referred to as generalized Mersenne prime. This form permits fast modular reduction. Solinas gave fast reduction algorithms for such the prime [4,15]. The following Algorithms 1, 2, 3, 4 and 5 show the dedicated reduction algorithms for  $p_{192}$ ,  $p_{224}$ ,  $p_{256}$ ,  $p_{384}$  and  $p_{521}$ , respectively.



---

**Algorithm 1** Fast reduction modulo  $p_{192} = 2^{192} - 2^{64} - 1$

---

**In** integer  $c = (c_5, c_4, c_3, c_2, c_1, c_0)$ , where each  $c_i$  is a 64-bit word, and  $0 \leq c < p_{192}^2$ .

**Out**  $c \bmod p_{192}$ .

1. **Define 192-bit integers:**

$$s_0 = (c_2, c_1, c_0)$$

$$s_1 = (0, c_3, c_3)$$

$$s_2 = (c_4, c_4, 0)$$

$$s_3 = (c_5, c_5, c_5)$$

2. **Return**  $s_0 + s_1 + s_2 + s_3 \bmod p_{192}$
- 

---

**Algorithm 2** Fast reduction modulo  $p_{224} = 2^{224} - 2^{96} + 1$

---

**In** integer  $c = (c_{13}, \dots, c_1, c_0)$ , where each  $c_i$  is a 32-bit word, and  $0 \leq c < p_{224}^2$ .

**Out**  $c \bmod p_{224}$ .

1. **Define 224-bit integers:**

$$s_0 = (c_6, c_5, c_4, c_3, c_2, c_1, c_0)$$

$$s_1 = (c_{10}, c_9, c_8, c_7, 0, 0, 0)$$

$$s_2 = (0, c_{13}, c_{12}, c_{11}, 0, 0, 0)$$

$$s_3 = (c_{13}, c_{12}, c_{11}, c_{10}, c_9, c_8, c_7)$$

$$s_4 = (0, 0, 0, 0, c_{13}, c_{12}, c_{11})$$

2. **Return**  $s_0 + s_1 + s_2 - s_3 - s_4 \bmod p_{224}$
- 

---

**Algorithm 3** Fast reduction modulo  $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

---

**In** integer  $c = (c_{15}, \dots, c_1, c_0)$ , where each  $c_i$  is a 32-bit word, and  $0 \leq c < p_{256}^2$ .

**Out**  $c \bmod p_{256}$ .

1. **Define 256-bit integers:**

$$s_0 = (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$$

$$s_1 = (c_{15}, c_{14}, c_{13}, c_{12}, c_{11}, 0, 0, 0)$$

$$s_2 = (0, c_{15}, c_{14}, c_{13}, c_{12}, 0, 0, 0)$$

$$s_3 = (c_{15}, c_{14}, 0, 0, 0, c_{10}, c_9, c_8)$$

$$s_4 = (c_8, c_{13}, c_{15}, c_{14}, c_{13}, c_{11}, c_{10}, c_9)$$

$$s_5 = (c_{10}, c_8, 0, 0, 0, c_{13}, c_{12}, c_{11})$$

$$s_6 = (c_{11}, c_9, 0, 0, c_{15}, c_{14}, c_{13}, c_{12})$$

$$s_7 = (c_{12}, 0, c_{10}, c_9, c_8, c_{15}, c_{14}, c_{13})$$

$$s_8 = (c_{13}, 0, c_{11}, c_{10}, c_9, 0, c_{15}, c_{14})$$

2. **Return**  $s_0 + 2s_1 + 2s_2 + s_3 + s_4 - s_5 - s_6 - s_7 - s_8 \bmod p_{256}$
-

---

**Algorithm 4** Fast reduction modulo  $p_{384} = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$

---

**In** integer  $c = (c_{23}, \dots, c_1, c_0)$ , where each  $c_i$  is a 32-bit word, and  $0 \leq c < p_{384}^2$ .

**Out**  $c \bmod p_{384}$ .

1. **Define 384-bit integers:**

$$s_0 = (c_{11}, c_{10}, c_9, c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$$

$$s_1 = (0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, 0, 0, 0, 0)$$

$$s_2 = (c_{23}, c_{22}, c_{21}, c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12})$$

$$s_3 = (c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{23}, c_{22}, c_{21})$$

$$s_4 = (c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{20}, 0, c_{23}, 0)$$

$$s_5 = (0, 0, 0, 0, c_{23}, c_{22}, c_{21}, c_{20}, 0, 0, 0, 0)$$

$$s_6 = (0, 0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, 0, 0, c_{20})$$

$$s_7 = (c_{22}, c_{21}, c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{23})$$

$$s_8 = (0, 0, 0, 0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, c_{20}, 0)$$

$$s_9 = (0, 0, 0, 0, 0, 0, 0, 0, c_{23}, c_{23}, 0, 0, 0)$$

2. **Return**  $s_0 + 2s_1 + s_2 + s_3 + s_4 + s_5 + s_6 - s_7 - s_8 - s_9 \bmod p_{384}$

---



---

**Algorithm 5** Fast reduction modulo  $p_{521} = 2^{521} - 1$

---

**In** integer  $c = (c_{1041}, \dots, c_1, c_0)$ , where  $c_i \in \{0, 1\}$ , and  $0 \leq c < p_{521}^2$ .

**Out**  $c \bmod p_{521}$ .

1. **Define 521-bit integers:**

$$s_0 = (c_{1041}, \dots, c_{522}, c_{521})$$

$$s_1 = (c_{520}, \dots, c_1, c_0)$$

2. **Return**  $s_0 + s_1 \bmod p_{521}$

---

## 2.2 Extra Reduction

Algorithms 1~5 may be executed with conditional branches in Step 2 depending on the input. A typical implementation of Step 2 in Algorithm 1 with  $p_{192}$  can be as the following.

2.1.  $t \leftarrow s_0 + s_1 + s_2 + s_3$

2.2. **While**  $t \geq p_{192}$  **do:**  $t \leftarrow t - p_{192}$  (extra reduction)

2.3. **Return**  $t$

To ensure the output of Algorithm 1 is smaller than  $p_{192}$ , when the intermediate value  $t$  in Step 2.1 becomes larger than or equal to the modulus  $p_{192}$ , we have to perform subtraction in Step 2.2. This conditional subtraction is called “*extra reduction*.” Extra reduction should be performed depending on the inputs to the modular multiplication. The extra reduction leads differences between running times needed for various input values. SPA using this information leakage will be discussed in Section 4.

### 2.3 Other Reduction Method for Modulus of Special Form

Standards for Efficient Cryptography Group (SECG) also provides recommended elliptic curve domain parameters over  $\mathbb{F}_p$  [14]. Their field order  $p$  also has special forms that permit fast implementation. Their special forms include the form  $p = 2^t - c$  for some small value of  $c$ , as well as generalized Mersenne prime.

An efficient reduction method for the modulus of the form  $p = 2^t - c$  is also well known [11]. A typical implementation for such modulus may include an extra reduction. Therefore, SPA, which will be discussed in Section 4, can be applied to the implementation of reduction with the special modulus  $p = 2^t - c$ .

## 3 Unified Code

Brier and Joye proposed unified code, that unifies the standard formulae for point addition and point doubling on Weierstrass form of an elliptic curve [3]. Unified code for projective coordinates, where a triplet  $(X, Y, Z)$  corresponds to the affine coordinates  $(X/Z, Y/Z)$  whenever  $Z \neq 0$ , is given in Algorithm 6.

The unified code can compute addition of two different points or doubling of a point with the same formulae. A unified approach offers generic solutions for preventing SPA. However, Walter pointed out that SPA may still be possible on the unified code combining with Montgomery multiplication if an extra reduction is included [18].

---

#### Algorithm 6 Unified point addition/doubling

---

**In**  $P_0 = (X_0, Y_0, Z_0), P_1 = (X_1, Y_1, Z_1) \in E(\mathbb{F}_p)$

**Out**  $P_2 = P_0 + P_1 = (X_2, Y_2, Z_2) \in E(\mathbb{F}_p)$

1.  $u_1 \leftarrow X_0 Z_1, u_2 \leftarrow X_1 Z_0, t \leftarrow u_1 + u_2$
  2.  $s_1 \leftarrow Y_0 Z_1, s_2 \leftarrow Y_1 Z_0, m \leftarrow s_1 + s_2$
  3.  $z \leftarrow Z_0 Z_2, f \leftarrow zm, l \leftarrow mf, g \leftarrow tl$
  4.  $r \leftarrow t^2 - u_1 u_2 + az^2, w \leftarrow r^2 - g$
  5.  $X_2 \leftarrow 2fw$
  6.  $Y_2 \leftarrow r(g - 2w) - l^2$
  7.  $Z_2 \leftarrow 2f^3$
  8. **Return**  $P_2 = (X_2, Y_2, Z_2)$
- 

## 4 Simple Power Analysis

In this section we discuss side channel leakage from the unified code (Algorithm 6) with fast implementation of modular reduction for NIST generalized Mersenne primes (Algorithms 1~5). We assume that the left-to-right binary method of elliptic point multiplication, shown in Algorithm 7, is used.

---

**Algorithm 7** Left-to-right binary method of elliptic point multiplication

---

**In**  $G \in E(\mathbb{F}_p)$  and  $k$ , where  $k = (k_{t-1}k_{t-2} \cdots k_0)_2$  and  $k_{t-1} = 1$

**Out**  $R = kG$

1.  $R \leftarrow G$
  2. **For**  $i$  **from**  $t - 2$  **down to**  $0$  **do**:
  3.      $R \leftarrow 2R$
  4.     **If**  $k_i = 1$  **then**  $R \leftarrow R + G$
  5.     **end for**
  6. **Return**  $R$
- 

#### 4.1 Probability of Extra Reduction

Firstly we evaluate the probability of extra reduction by our experiments. In this evaluation we assume the following.

- Two inputs  $a$  and  $b$  to modular multiplication  $c \leftarrow a \times b \bmod p$  are uniformly distributed modulo  $p$ .
- Modular multiplication is carried out as follows: firstly compute  $c \leftarrow a \times b$ , then perform modular reduction  $c \leftarrow c \bmod p$  using Algorithm 1~5.
- Step 2 in Algorithm 1~5 is performed as illustrated in Section 2.2. That is, first add (or subtract)  $s_i$  for all  $i$ , next if the intermediary result  $t$  is greater than or equal to modulus  $p$ ,  $p$  should subtract from  $t$ .

The probability of extra reduction depends on the inputs  $a$  and  $b$  to the modular multiplication  $c \leftarrow a \times b \bmod p$ . The following three cases should be considered: 1) The case that  $a$  and  $b$  are distinct. 2) The case that  $a = b$ . 3) The case that one of  $a$  and  $b$  is a constant.

In the third case,  $a$  or  $b$  is constant, the probability of extra reduction can be evaluated depending on the value of constant.

We should notice that in signature generation operation of Elliptic Curve Digital Signature Algorithm (ECDSA), the left-to-right method of elliptic point multiplication performs point addition with fixed point  $G$ , which is a base point on an elliptic curve (Step 4 of Algorithm 7). Therefore, when point addition with two different points are computed, in the first two steps of the unified code (Step 1 and 2 in Algorithm 6), modular multiplication should be performed with a constant value ( $x$ ,  $y$  or  $z(= 1)$  coordinates of the base point  $G$ ).

**Random Input.** Table 1 shows the probability of extra reduction in the two cases: 1) two inputs to modular multiplication are different, 2) two inputs are the same. We have carried out an experiment as follows. We generated 100,000 pairs of integers  $a$  and  $b$  ( $0 \leq a, b < p$ ), then we examined whether the extra reduction was performed or not. SHA-1 based pseudo random number generator described in FIPS 186-2 [2] was used to generate integers.

**Table 1.** Probability of extra reduction with random inputs, where  $p$  is a NIST recommended domain parameter

Curve	$a \times b \pmod p, (a \neq b)$	$a \times a \pmod p$
P-192	0.69	0.73
P-224	0.30	0.27
P-256	0.11	0.20
P-384	0.65	0.69
P-521	0.25	0.33

**Table 2.** Probability of extra reduction with fixed base point  $G = (x_G, y_G, 1)$ , where  $p$  and  $G$  are NIST recommended domain parameters

Curve	$a \times x_G \pmod p$	$a \times y_G \pmod p$
P-192	0.54	0.51
P-224	0.22	0.22
P-256	0.04	0.02
P-384	0.70	0.57
P-521	0.19	0.28

We can observe from Table 1 that in the case of P-192 and P-384, the probability of extra reduction is significantly large compared to other curves. As we can see from Step 2 in Algorithms 1~5, in the case of P-192, all  $s_i$  should be added. In the case of P-384, relatively large number of  $s_i$  should be added. So the probability can be large in these two curves. On the other hand, in the cases of P-224, P-256 and P-521, relatively small number of  $s_i$  should be added. Consequently, the probability of the extra reduction should be small. P-256 has smallest probability of extra reduction in consequence of large number of  $s_i$  which should be subtracted.

**Base Point.** Table 2 shows the probability of extra reduction in the case that one input to the modular multiplication is  $x$  coordinate or  $y$  coordinate of the NIST recommended base point. As in the case of random inputs, we randomly generated 100,000 integers  $a$  ( $0 \leq a < p$ ), then examined the probability.

The value of  $x$  coordinate and  $y$  coordinate of the NIST base points [2] are given below in hexadecimal format. Upper digits are shown because of the space limitation.

- P-192:**  $x_{192} = 188da80eb03090f67cbf20eb43a18800f4f0 \dots$   
 $y_{192} = 07192b95ffc8da78631011ed6b24cdd573f9 \dots$
- P-224:**  $x_{224} = b70e0cbd6bb4bf7f321390b94a03c1d356c2 \dots$   
 $y_{224} = bd376388b5f723fb4c22dfe6cd4375a05a07 \dots$
- P-224:**  $x_{224} = b70e0cbd6bb4bf7f321390b94a03c1d356c2 \dots$   
 $y_{224} = bd376388b5f723fb4c22dfe6cd4375a05a07 \dots$
- P-256:**  $x_{256} = 6b17d1f2e12c4247f8bce6e563a440f27703 \dots$   
 $y_{256} = 4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce \dots$

**P-384:**  $x_{384} = \text{aa87ca22be8b05378eb1c71ef320ad746e1d} \dots$   
 $y_{384} = \text{3617de4a96262c6f5d9e98bf9292dc29f8f4} \dots$

**P-521:**  $x_{521} = \text{0c6858e06b70404e9cd9e3ecb662395b4429} \dots$   
 $y_{521} = \text{11839296a789a3bc0045c8a5fb42c7d1bd99} \dots$

We can observe from Table 2 that in the case of P-192, the probability of extra reduction is smaller than the case of random inputs (Table 1). The reason is that the most significant digits of  $x$  and  $y$  coordinates of the base point have small value, 1 for  $x$  and 0 for  $y$ . Therefore, the most significant word ( $c_5$  in Algorithm 1) have smaller value with high probability. Consequently, with high probability,  $s_0 + s_1 + s_2 + s_3$  can have smaller value than  $p_{192}$ .

On the other hand, in the case of P-224, although the most significant digits of both  $x$  and  $y$  coordinates have the value  $0xb$ , which is relatively large, the probability of the extra reduction is small. We can notice from Algorithm 2 that the most significant word  $c_{13}$  is not included in the most significant word of  $s_0$ ,  $s_1$  and  $s_2$  (plus terms), and is included only in the most significant word of  $s_3$  (minus term). Consequently, the probability of extra reduction become smaller when an input to the modular reduction has large value.

We can observe the same facts as P-192 and P-224 in the curves P-256, P-384 and P-521. We will discuss SPA on the modular reduction in next section. We will show that, in the case of one of inputs to the modular multiplication is  $x$  or  $y$  coordinate of the base point, an implementation of elliptic point multiplication with unified code should be susceptible to SPA if the probability of the extra reduction is large.

We will see in the later section that, in the setting of ECDSA, the probability of the extra reduction with fixed base point strongly affects the attack, which will be described in the next section. Therefore, a theoretical analysis of the probability of the extra reduction is useful to give strict evaluation of security against the attack.

## 4.2 The Attacks

Walter proposed SPA against elliptic point multiplication with the unified code and Montgomery multiplication including extra reduction [18]. In this section, we will discuss SPA against dedicated fast reduction method for NIST recommended domain parameters. We assume that implementation of elliptic point multiplication is based on the following.

- Algorithms 1~5 are used for dedicated fast modular reduction for NIST recommended domain parameters. The algorithm described in Section 2.2 is also used for a sub-routine.
- The unified code, Algorithm 6, is used for elliptic point addition and doubling.
- The left-to-right method, Algorithm 7, is used for the elliptic point multiplication.

We also assume the model of the attacker as follows.

**Table 3.** An Attack on P-192

$k$	1001 01 1 001 001 001 01 1 01 1 01 1 1 1 0001 01 1 01 ...
$\mathcal{AD}$	DDDDADADADDADDADDADDADDADADDADADDADADADADDADDADDADD ...
$u_1$	-----*-*-*-*-*-----*-*-*-----*-*-----*-*-----*-* ...
$u_2$	-----*-*-*-*-*-----*-*-----*-*-----*-*-----*-*-----*-* ...
$s_1$	-----*-*-----*-*-----*-*-----*-*-----*-*-----*-*-----*-* ...
$s_2$	-----*-*-----*-*-----*-*-----*-*-----*-*-----*-*-----*-* ...
Difference	-----Y-Y---Y---Y-----Y---Y-Y--Y-----Y---Y---Y--- ...
Recovered $\mathcal{AD}$	-----YYYYY-YYY-YYY-----YYY--YYYYYYYYY-----YYY--YYY--YYYYY ...

- The attacker has access to a device which calculates elliptic point multiplication.
- The attacker has knowledge about the implementation.
- The attacker can distinguish the extra reduction in Algorithms 1~5 by monitoring power consumption during the computation of the elliptic point multiplication.

When two inputs  $P_0$  and  $P_1$  to elliptic point addition are the same (i.e.  $X_0 = X_1, Y_0 = Y_1, Z_0 = Z_1$ ), the computation of  $u_1$  and  $u_2$  in Step 1 of the unified code (Algorithm 6) should be identical. The computation of  $s_1$  and  $s_2$  (Step 2) should also be identical. Therefore, both pairs exhibit identical behavior with respect to the occurrence of the extra reduction. It is the repeated or different behavior within the pairs which creates the handle for an attack. If the recorded behavior is different at these points, the point operation must be a point addition [18]. The attacker should guess the corresponding bit of the secret exponent to be “1” in such the case.

### 4.3 Example of the Attack

This section demonstrates an attack on P-192. Table 3 shows a typical example of the attack by simulation on PC. This table shows the initial few bits.  $k$  is the randomly generated secret exponent to be recovered, which is represented in binary format. “ $\mathcal{AD}$ ” denotes the sequence of point addition (A) and point doubling (D) in the left-to-right binary method of elliptic point multiplication (Algorithm 7). The symbol “\*” expresses that the extra reduction has been revealed during the computation of  $u_1, u_2, s_1$  and  $s_2$ . The symbol “-” expresses that the extra reduction has not been revealed.

The penultimate row records difference (marked Y) within the computation of the pair  $(u_1, u_2)$  and the pair  $(s_1, s_2)$ . In the place at marked Y, the corresponding elliptic point operation must be addition of two different points, then the attacker must guess that the corresponding bit of the secret exponent is “1”. The bottom row “Recovered  $\mathcal{AD}$ ” expresses that executed elliptic point operation (A or D) has been successfully recovered (marked Y). We should notice that once a difference was observed (at the penultimate row), these must be all point addition, then point doubling must be performed on either side of a known point addition.

In this simulation, the point  $P_1$  in the unified code (Algorithm 6) was regarded as the fixed point  $G$  in the binary method (Algorithm 7). In ECDSA, the fixed point  $G$  is the domain parameter. In a typical implementation,  $G$  is represented in affine coordinates. That is  $Z$  coordinate is always 1. In such the case, the extra reduction never be performed within the computation of  $u_1$  and  $s_1$ . Therefore, if the probability of the extra reduction is large, the large number of bits in the secret exponent should be recovered. As we have already mentioned, the probability of the extra reduction depends on the value of  $x$  and  $y$  coordinates.

We have a further work on a theoretical estimation of the probability of successfully recovered bits. As we have mentioned in the previous section, a theoretical analysis of the probability of the extra reduction may useful for this estimation.

#### 4.4 Countermeasure

Even if the unified code is used for elliptic point operation, underlying modular operation should also be unified. That is, a conditional operation, which depends on the inputs, can constitute an information channel, providing the attacker with valuable information on the secret exponent.

As we have already stated, the probability of the extra reduction and the number of successfully recovered bits depend on the value of coordinates of the base point. If we can generate appropriate base point in terms of our attack, the probability of the extra reduction would be small. However, this strategy may not be an essential solution of countermeasure.

We should notice that, as Walter stated in [18], Coron's three countermeasures against differential power analysis (DPA) [5] are insufficient. The first countermeasure: randomization of the secret exponent, can not defeat the attack when the attacker can distinguish the extra reduction from one power trace. The second countermeasure: blinding the point, and the third countermeasure: randomized projective coordinates also can not be countermeasures. Changing coordinates value may affect the probability of the extra reduction, but does not defeat the attack.

## 5 Concluding Remarks

In this paper we showed SPA on the fast modular reduction with generalized Mersenne prime included in NIST recommended domain parameters, combining with the unified code for elliptic point operations. A practical cryptosystem based on an elliptic curve over  $\mathbb{F}_p$  is implemented with such a special form of prime to achieve fast implementation. A typical implementation of a dedicated modular reduction method for a special prime includes extra reduction. Such the conditional operation can give side channel leakage.

When we take a unified approach to prevent simple power attacks, it is not sufficient to take into account only elliptic point operation level. Careful implementation at lower level, i.e. modular operation level, should be considered.



## References

1. 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, pp.175–190, Springer-Verlag, 1998.
2. “Digital signature standard (DSS),” FIPS PUB 186-2, U.S. National Institute of Standards and Technology, 2000.
3. E. Brier and M. Joye, “Weierstrass elliptic curves and side-channel attacks,” Public Key Cryptography – PKC 2002, LNCS 2274, pp.335–345, Springer-Verlag, 2002.
4. M. Brown, D. Hankerson, J. Lopez and A. Menezes, “Software implementation of the NIST elliptic curves over prime fields,” Technical Report CORR 2000-56, University of Waterloo, 2000.
5. J.-S. Coron, “Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems,” Cryptographic Hardware and Embedded Systems – CHES’99, LNCS 1717, pp.292–302, Springer-Verlag, 1999.
6. C. Gebotys and R. Gebotys, “Secure elliptic curve implementations: an analysis of resistance to power-attacks in a DSP processor,” Cryptographic Hardware and Embedded Systems – CHES 2002, LNCS 2523, pp.114–128, Springer-Verlag, 2002.
7. G. Hachez and J.-J. Quisquater, “Montgomery exponentiation with no final subtractions: Improved Results,” Cryptographic Hardware and Embedded Systems – CHES 2000, LNCS 1965, pp.293–301. Springer-Verlag, 2000.
8. M. Joye and J.-J. Quisquater, “Hessian elliptic curves and side channel attacks,” Cryptographic Hardware and Embedded Systems – CHES 2001, LNCS 2162, pp.402–410, Springer-Verlag, 2001.
9. P.C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” Advances in Cryptology – CRYPTO ’96, LNCS 1109, pp.104–113, Springer-Verlag, 1996.
10. P.-Y. Liardet and N.P. Smart, “Preventing SPA/DPA in ECC systems using the Jacobi form,” Cryptographic Hardware and Embedded Systems – CHES 2001, LNCS 2162, pp.391–401, Springer-Verlag, 2001.
11. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, “Handbook of applied cryptography,” CRC Press, 1996.
12. P.L. Montgomery, “Modular multiplication without trial division,” Mathematics of Computation, vol. 44, no. 170, pp.519–521, 1885.
13. W. Schindler, “A timing attack against RSA with the Chinese Remainder Theorem,” Cryptographic Hardware and Embedded Systems – CHES 2000, LNCS 1965, pp.109–124, Springer-Verlag, 2000.
14. Certicom, “Standard for efficient cryptography, SEC2: Recommended elliptic domain parameters,” 2000.
15. J.A. Solinas, “Generalized Mersenne numbers,” Technical Report CORR 99-39, University of Waterloo, 1999.
16. C.D. Walter, “Montgomery exponentiation needs no final subtractions,” Electric Letters, vol. 35, no. 21, pp.1831–1832, 1999.
17. C.D. Walter and S. Thompson, “Distinguishing exponent digits by observing modular subtractions,” RSA Conference 2001, LNCS 2020, pp.192–207, Springer-Verlag, 2001.
18. C.D. Walter, “Simple power analysis of unified code for ECC double and add,” Cryptographic Hardware and Embedded System – CHES 2004, LNCS 3156, pp.191–204, Springer-Verlag, 2004.

# Asymmetric Concurrent Signatures

Khanh Nguyen

Gemplus R&D,  
12 Ayer Rajah Crescent,  
Singapore 139941  
kenny.nguyen-qk@gemplus.com

**Abstract.** The concept of concurrent signatures allows two entities to produce two signatures in such a way that, the signer of each signature is ambiguous from a third party's point of view until the release of a secret, known as the *keystone*. Once the keystone is released, both signatures become binding to their respective signers concurrently. Previous concurrent signature schemes use the concept of ring signatures in their construction. Ring signatures identify the ring and thus concurrent signatures constructed from ring signature are related and linkable. We propose a new concurrent signature scheme which is independent of the ring signature concept. Our concurrent signatures are anonymous. The ordinary signatures obtained from our concurrent signature protocol are unlinkable and do not reveal which concurrent signature transaction has occurred. The price we pay is our concurrent signatures are *asymmetric* in the sense that the initial signature and subsequent signatures are not of the same construction.

## 1 Introduction

### 1.1 Concurrent Signatures

The concept of concurrent signatures was introduced in [3]. In a concurrent signature scheme, two parties  $A$  and  $B$  interact without the help of any third party to sign messages  $M_A$  and  $M_B$  in such a way that both signatures are ambiguous without an extra piece of information, known as the *keystone*. Without the keystone, from a third party's point of view, both signatures are ambiguous as they could have been generated by either of the parties and thus do not represent any entity's commitments to the messages. With the keystone, the signer for each signature is identified and both signatures become instantly binding to their respective signers. In the original proposal [3], the keystone is a randomly chosen piece of information and possessed by the protocol's initiator. During the signature generation phase, the keystone is not known to other parties. When the validation of concurrent signatures is required, the initiator reveals the keystone which validates all the signatures *concurrently*.

Concurrent signatures find applications in fair exchange of digital signatures and fair tendering of contracts. To exchange digital signatures, two parties  $A$  and  $B$  engage in a concurrent signature protocol by which each party receives

the other's concurrent signature on the desired message. The exchange is fair in the sense that without the knowledge of the keystone, both of the concurrent signatures are ambiguous and non-binding to the signers. To prove the validity of the concurrent signature generated by party  $B$ , party  $A$  would need to reveal the keystone. The revelation of this knowledge in turn would bind the concurrent signature generated by party  $A$  to its signer, i.e., to party  $A$ , hence the fairness is achieved. This notion of fairness is weak as it allows party  $A$  to reveal the keystone in private to another party. In this scenario, the party  $B$  who is denied of the knowledge of the keystone, can not prove the validity of the concurrent signature generated by party  $A$ . However, this weak notion of fairness is acceptable in many instances, such as contracts for party  $A$  to buy physical goods from party  $B$ . The advantage of fair exchange using concurrent signatures is that it eliminates the requirement of a trusted third party from the protocol and it is not overly interactive. Previous solutions to the problem of fair exchange of digital signatures (see [1,2] and references therein for a detailed survey) are either highly interactive with multiple rounds of exchange or require the existence of a third party trusted by both parties  $A$  and  $B$ . Multiple rounds of exchange are inefficient while the existence of a trusted third party is not always warranted.

Previous concurrent signature proposals [3,10] are based on the concept of ring signatures. In those schemes, each concurrent signature is a ring signature [8] generated from the ring consisting of all involved parties. Each ring signature while identifies that the signer is a member of the ring, does not reveal the actual signer. The ambiguity of each concurrent signature comes from this anonymous attribute of the underlying ring signature scheme. Ring signatures while hide the actual signer identity in the ring, do identify the ring. Hence it does leak information about the transaction.

Furthermore, in those proposals [3,10], once the keystone is revealed, its value must be included in each signature in order to bind the signature to its actual signer. This means all the signatures obtained from a concurrent signature protocol are linked together. This is not desirable when the anonymity of the transaction is required.

## 1.2 Our Contributions

In this paper, we propose a concurrent signature scheme which offers a stronger notion of anonymity than previous schemes. Under the Decisional Diffie-Hellman assumption, each of our concurrent signatures could be created by anyone (not just by a member of the ring as done previously) without the knowledge of the keystone. Due to this, a set of  $k$  concurrent signatures in our scheme is *anonymous* and releases no information about whether the transaction is real or faked. Once the keystone is revealed, our concurrent signatures could be converted to ordinary Schnorr-like signatures. Except for one signature issued by the protocol initiator, our converted signatures do not contain the keystone and thus remain *unlinkable*. This is in total contrast to all previous proposals [3,10] in which once the keystone is revealed, all the signatures obtained from a concurrent signature transaction are linked by the formation of the ring.

The cost we pay is that our concurrent signatures are *asymmetric*. The signature signed by the party who processes the keystone is different from the other signatures. However, this is not a real issue in most applications where concurrent signatures find their uses.

### 1.3 Technical Approach

Each concurrent signature could be viewed as a *promise of signature* which could be converted into an ordinary signature by revealing a secret information – the *keystone*. In CKP concurrent signatures, a promise of signature is a ring signature issued by a member of the ring. Upon the release of the keystone information, it converts the ring signature to a signature that could only be issued by the identified member. While ring signatures could be used to construct promises of signatures, it is not a necessary condition.

In [1], Asokan, Shoup and Waidner (ASW) proposed a technique which reduces a promise of a signature to a promise of a homomorphic pre-image. Without the homomorphic pre-image, the promise of signature looks indistinguishable from random elements of the signature space as it could have been created by anyone using solely public information. With the homomorphic pre-image, the promise of signature could be converted into an ordinary signature. This technique is applicable to most well-known signature schemes (e.g., Schnorr [9], RSA [7], DSS [5] and GQ [4] signature schemes). If we view the homomorphic pre-image as the keystone, then an ASW promise of signature is indeed a concurrent signature. The advantage of using the ASW construction is that a promise of signature while contains public information about a single entity, could be created by anyone; hence without the keystone, it leaks no information about the signature. This is a genuine advantage from ring signatures where signatures are linked together by the nature of the ring.

An ASW promise of a signature could only be constructed with the knowledge of homomorphic pre-image, i.e., the keystone. In the standard model of the concurrent signatures, the keystone should only be known to one party and not to be shared with other parties. Thus except for one party in the protocol, the ASW promise of signature construction could not be used for others. In those cases, we need a promise of signature construction that does not require the knowledge of the keystone. Fortunately, a such new construction is possible for a variant of the Schnorr signature scheme. This construction is described in section 2.

### 1.4 Organization

The remaining of the paper is organized as follows. Section 2 describes the concept of promises of signatures. This concept forms the main basic building block for our concurrent signature scheme. Section 3 discusses our formal definitions which include both the definitions for concurrent signature scheme and protocol. Section 4 gives the security model for our concurrent signature scheme. Here, we modify the security model from [3] in order to capture the additional security

properties of *anonymity* and *unlinkability*. Section 5 proposes a concrete concurrent signature scheme. It also discusses the security of the construction and extensions of this proposal to multi-party case.

## 2 Promises of Signatures

**Definition 1.** *Let  $f$  be some cryptographic function. The value  $\sigma = \langle s, u \rangle$  is said to be a valid promise of signature  $\rho = \langle k, u \rangle$  on some message  $m$  if the following conditions hold:*

- **Publicly Verifiable:** *given  $\sigma$ , everyone is convinced that if there exists  $k = f^{-1}(s)$  then  $\rho = \langle k, u \rangle$  is a valid ordinary signature.*
- **Anonymity:** *without the knowledge of  $k = f^{-1}(s)$ ,  $\sigma$  is indistinguishable from random elements of the signature space.*

To convert a promise of signature to an ordinary signature, one only needs to reveal the value  $k$  such that  $f(k)$  gives  $s$ . The verification  $s = f(k)$  is adequate to verify the signature  $\rho = \langle k, u \rangle$  provided that  $\sigma = \langle s, u \rangle$  is a valid promise of signature.

### 2.1 Promises of Schnorr Signatures

The Schnorr signature scheme [9] is constructed as follows:

- **Setup:** the public parameters are the primes  $p$  and  $q$  of appropriate size such that  $q|(p-1)$ , and a generator  $g$  for the subgroup in  $\mathbb{Z}_p^*$  of order  $q$ .
- **Key Generation:** select a random  $x \in \mathbb{Z}_q$  and compute  $h = g^x \bmod p$ . The public key is  $\{p, q, g, h\}$  and its corresponding private key is  $x$ .
- **Sign:** the signer chooses a random  $r \in \mathbb{Z}_q$  and computes  $k = cx + r \bmod q$  where  $c = H(g^r \bmod p, m)$  and  $H$  is a hash function. The signature is  $\rho = (k, c)$ .
- **Verify:** to verify the signature  $\rho = (k, c)$ , one checks  $c = H(g^k h^{-c} \bmod p, m)$ .

For the remaining of this paper, we omit the modular reduction  $\bmod p$  from the modular exponentiation  $a^r \bmod p$  for any given  $a \in \mathbb{Z}_p^*$  and any arbitrary  $r$ . Instead, we use  $a^r$  to denote  $a^r \bmod p$  whenever the context is clear.

The promise of the Schnorr signature  $\rho = \langle k, c \rangle$  is  $\sigma = \langle s, c \rangle$ , where  $s = f(k)$  for  $f(x) = g^x$ . To verify the promise of signature, one verifies  $c = H(sh^{-c}, m)$ . To convert the promise of signature to an ordinary signature, the signer reveals  $k$ . The verification is  $g^k = s$  which implies  $c = H(g^k h^{-c}, m)$ , and thus  $\rho = \langle k, c \rangle$  is a valid Schnorr signature.

**Lemma 1.** *The value  $\sigma = \langle s, c \rangle$  where  $c = H(sh^{-c}, m)$  is a promise of signature  $\rho = \langle k, c \rangle$ , where  $s = f(k)$  for  $f(x) = g^x$ .*

## 2.2 Promises of Schnorr-Like Signatures

Apart from the above promise of signature. We also need a promise of signature construction in which the promise of signature could be constructed without the knowledge of the promise, i.e., without the knowledge of the homomorphic inverse. We design such a promise of signature construction for a variant of Schnorr signature scheme. This variant of Schnorr signature scheme is constructed as follows:

- **Setup:** the public parameters are the primes  $p$  and  $q$  of appropriate size such that  $q|(p-1)$ , and a generator  $g$  for the subgroup in  $\mathbb{Z}_p^*$  of order  $q$ .
- **Key Generation:** select a random  $x \in \mathbb{Z}_q$  and compute  $h = g^x$ . The public key is  $\{p, q, g, h\}$  and the corresponding private key is  $x$ .
- **Sign:** the signer chooses two random  $\kappa$  and  $r \in \mathbb{Z}_q$  and computes  $\kappa = (r-c)/x \bmod q$  where  $c = H(g^r, m)$  and  $H$  is a hash function. The signature is  $\rho = (\kappa, c)$ .
- **Verify:** to verify the signature  $\rho = (\kappa, c)$ , one checks  $c = H(g^c h^\kappa, m)$ .

This is a straightforward variant of the Schnorr signature scheme. The difference here is that the equation  $r = \kappa x + c \bmod q$  is used instead of the standard equation  $r = k - cx \bmod q$ . Using essentially the same security proof for the original Schnorr signature scheme [6], we have the following security result:

**Lemma 2.** *In the random oracle model, this Schnorr-like signature scheme is existentially unforgeable against adaptive chosen message attacks, assuming the hardness of the discrete logarithm problem.*

The promise of the Schnorr-like signature  $\rho = \langle \kappa, c \rangle$  is  $\omega = \langle s, \kappa_1, c \rangle$  where  $s = f(\kappa - \kappa_1)$  for  $f(x) = h^x$ . To verify this promise of signature, one verifies  $c = H(g^c h^{\kappa_1} s, m)$ . To convert the promise of signature to an ordinary signature, the signer reveals  $\kappa_2 = \kappa - \kappa_1 \bmod q$ . The verification is  $h^{\kappa_2} = s$  which implies  $c = H(g^c h^{\kappa_1 + \kappa_2}, m)$ ; and hence  $\omega = \langle \kappa, c \rangle = \langle \kappa_1 + \kappa_2 \bmod q, c \rangle$  is a valid signature. Note that the knowledge of  $\kappa_2$  is not *required* in order to create the promise of signature  $\omega$ .

**Lemma 3.** *The value  $\omega = \langle s, \kappa_1, c \rangle$  where  $c = H(g^c h^{\kappa_1} s, m)$  is a valid promise of signature  $\rho = \langle \kappa, c \rangle$ , where  $\kappa = \kappa_1 + \kappa_2 \bmod q$  and  $s = f(\kappa_2)$  for  $f(x) = h^x$ .*

*Proof.* The verification  $c = H(g^c h^{\kappa_1} s, m)$  requires only public information. If there exists  $\kappa_2$  such that  $h^{\kappa_2} = s$ , we have  $c = H(g^c h^{\kappa_1 + \kappa_2}, m)$  and thus  $\rho = \langle \kappa, c \rangle = \langle \kappa_1 + \kappa_2, c \rangle$  is a valid ordinary signature. Hence, the publicly verifiable condition is satisfied.

The anonymity property of this promise of signature follows from the fact that one could generate a valid promise of signature from public information: the simulator chooses two random  $r$  and  $\kappa_1 \in \mathbb{Z}_q$ , computes  $c = H(g^r h^{\kappa_1}, m)$  and sets  $s = g^{r-c}$ . We have  $\omega = \langle s, \kappa_1, c \rangle$  satisfies  $c = H(g^c h^{\kappa_1} s, m)$  and thus is a valid promise of signature constructed solely from public information.

### 3 Formal Definitions

#### 3.1 Asymmetric Concurrent Signatures

In this section, we give a formal definition for concurrent signature schemes. Our definition is an adaption of the CKP definition with some modification accommodating the asymmetric property of signatures.

**Definition 2.** *An asymmetric concurrent signature scheme is a digital signature scheme consisted of the following algorithms:*

*SETUP:* A probabilistic algorithm that on input of a security parameter  $l$ , outputs: the set of participants  $\mathcal{U}$  each equipped with a private key  $x_i$  and the corresponding public key  $X_i$ . The algorithm also outputs the message space  $\mathcal{M}$ , the keystone space  $\mathcal{K}$ , the keystone fix space  $\mathcal{F}$  and some additional system parameters  $\pi$ . The algorithm also defines a function  $KGEN : \mathcal{K} \rightarrow \mathcal{F}$ , a set of functions  $\{KGEN_j : \mathcal{K} \rightarrow \mathcal{F}\}$  and a keystone transformation function  $KTRAN : \mathcal{F} \times \{x_i\} \rightarrow \mathcal{F}$ .

*ISIGN:* A probabilistic algorithm that on inputs  $\langle X_i, x_i, M_i \rangle$ , where  $X_i$  is a public key,  $x_i$  is the corresponding private key and  $M_i \in \mathcal{M}$  is the message, outputs a promise of signature  $\sigma_i = \langle s, u_i \rangle$  and the relevant keystone  $k \in \mathcal{K}$ , where  $s = KGEN(k)$ .

*SSIGN:* A probabilistic algorithm that on inputs  $\langle X_j, x_j, M_j \rangle$  and  $s$ , where  $X_j$  is a public key,  $x_j$  is the corresponding private key,  $M_j \in \mathcal{M}$  is the message and  $s$  a keystone fix, outputs a promise of signature  $\omega_j = \langle s', v_j \rangle$  where  $s' = KTRAN(s, x_j)$ .

*IVERIFY:* An algorithm which takes as input a promise of signature  $\sigma_i = \langle s, u_i \rangle$  and a message  $M_i$ , and outputs accept or reject. The algorithm outputs accept if and only if  $\sigma_i$  is a valid promise of the signature  $\langle k, \sigma_i \rangle$  on message  $M_i$  with  $f(x) = KGEN(x)$ .

*SVERIFY:* An algorithm which takes as input a promise of signature  $\omega_j = \langle s', v_j \rangle$ , a message  $M_j$ , and outputs accept or reject. The algorithm outputs accept if and only if  $\omega_j$  is a valid promise of the signature  $\langle k, \omega_j \rangle$  on message  $M_j$  with  $f(x) = KGEN_j(x)$ .

*VERIFY:* An algorithm which takes as input, a promise of signature  $\sigma_i = \langle s, u_i \rangle$  (or  $\omega_j = \langle s', v_j \rangle$ ), a message  $M$  and a keystone  $k \in \mathcal{K}$ , and outputs accept or reject. The algorithm will output accept if and only if  $s = KGEN(k)$  and  $\langle k, \sigma_i \rangle$  (or  $s' = KGEN_j(k)$  and  $\langle k, \omega_j \rangle$ ) forms a valid signature on message  $M$ .

We note that our concurrent signature protocol differs from the protocol of [3] in that the keystone fixes used for  $A$  and  $B$  are different. This crucial property allows us to unlink signatures obtained from the same signature protocol. For this to work, we require the keystone fix transform function  $KTRAN$  to be isomorphic and that  $s'$  leaks no more information about  $k$  rather than  $s$ . This security notion is discussed in the following section.

### 3.2 Concurrent Signature Protocol

Using the above definition, our two-party concurrent signature protocol is as follows. Let  $A$  be the initial signer who initiates the protocol and  $B$  be the matching signer who responds to the initial signer.  $A$  and  $B$  first run SETUP algorithm to generate the public key parameters for the system. We assume  $A$ 's public and private keys are  $X_A$  and  $x_A$  respectively. Likewise,  $B$ 's public and private keys are  $X_B$  and  $x_B$  respectively. Then  $A$  and  $B$  engage in the following protocol:

1: To start of the protocol,  $A$  runs ISIGN algorithm to generate a promise of signature  $\sigma_A = \langle s, u_A \rangle$  and the relevant keystone  $k$  for the message  $M_A$ . The values of  $\sigma_A$  and  $M_A$  are sent to  $B$ .

2: Upon receiving  $\sigma_A$  and  $M_A$ ,  $B$  verifies the validity of  $\sigma_A$  using IVERIFY. If  $\sigma_A$  is a valid promise of signature on message  $M_A$ ,  $B$  uses the keystone fix  $s$  to run SSIGN algorithm to generate a promise of signature  $\omega_B = \langle s', v_B \rangle$  for the message  $M_B$ . The values of  $\omega_B$  and  $M_B$  are sent to  $A$ .

3: Upon receiving  $\omega_B$  and  $M_B$ ,  $A$  runs SVERIFY to verify the validity of  $\omega_B$ . If so,  $A$  uses the keystone  $k$  to verify the keystone fix  $s'$ . If this keystone fix is valid,  $A$  forwards the keystone  $k$  to  $B$ .

## 4 Security Model

The original security model of [3] addresses four basic security properties, namely completeness, fairness, ambiguity and unforgeability. The unforgeability property is somewhat redundant as it has already been captured with the fairness property. If concurrent signatures are forgeable, fairness would not be achieved.

We require our concurrent signature protocol to achieve four security requirements, namely completeness, fairness, anonymity and unlinkability. The completeness and fairness properties are the standard security requirements in the original model of [3] while the anonymity and unlinkability properties are new. Our anonymity property replaces the ambiguity property in the original model. It provides a stronger anonymity attribute for concurrent signatures than what was originally allowed with the ambiguity property. The unlinkability property addresses the anonymity of ordinary signatures obtained from the concurrent signature protocol. This property is new and was not addressed in the original model.

We note that our anonymity and unlinkability properties are not overlapped. The anonymity property deals with the concurrent signatures while the unlinkability property deals with the ordinary signatures obtained from the concurrent signature protocol (upon the revelation of the keystone). It is quite possible that once the keystone is revealed, it would link the anonymous concurrent signatures together and thus the obtained (ordinary) signatures are not unlinkable.



#### 4.1 Completeness

**Definition 3.** A concurrent signature protocol is complete if the following conditions hold:

- If  $\sigma_i = \langle s, u_i \rangle = ISIGN(X_i, x_i, M_i)$ , then  $IVERIFY(s, u_i) = \text{accept}$ . Moreover, if  $KGEN(k) = s$  for some  $k \in \mathcal{K}$ ,  $VERIFY(k, s, u_i) = \text{accept}$ .
- If  $\omega_j = \langle s', v_j \rangle = SSIGN(X_j, x_j, M_j)$ , then  $SVERIFY(k, s', v_j) = \text{accept}$ . Moreover, if  $KGEN_j(k) = s'$  for some  $k \in \mathcal{K}$ ,  $VERIFY(k, s', v_j) = \text{accept}$ .
- If  $KGEN(k) = s$  and  $KGEN_j(k') = s'$ , then  $k = k'$ .

#### 4.2 Fairness

The fairness property is defined by the following game between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ .

GAME 1:

**Setup:** as SETUP in section 3.1. The challenger  $\mathcal{C}$  is given the public key  $X$  and its corresponding secret key  $x$ . The adversary is given the public key  $X'$  and its corresponding secret key  $x'$ .

**Queries:** the adversary  $\mathcal{A}$  is allowed to make a sequence of the following queries to the challenger  $\mathcal{C}$ :

- *ISIGN* query,  $\mathcal{A}$  will supply the message  $m_i$  and  $\mathcal{C}$  will output the promise of signature  $\sigma_i = \langle s_i, u_i \rangle = ISIGN(X, x, m_i)$ .
- *SSIGN* query,  $\mathcal{A}$  will supply his promise of signature  $\hat{\sigma}_j = \langle \hat{s}_j, \hat{u}_j \rangle$  with the message  $m_j$ . If  $IVERIFY(\hat{s}_j, \hat{u}_j) = \text{accept}$ ,  $\mathcal{A}$  obtains from  $\mathcal{C}$ :  $\omega_j = \langle s_j, v_j \rangle = SSIGN(X, x, m_j, \hat{s}_j)$ ; otherwise  $\mathcal{A}$  obtains nothing.
- *KRELEASE* query,  $\mathcal{A}$  requests  $\mathcal{C}$  to reveal the keystone  $k \in \mathcal{K}$  used to produce the keystone fix  $s \in \mathcal{F}$  in a previous *ISIGN* query.

**Output:** Finally  $\mathcal{A}$  outputs a string  $\phi$  and  $\mathcal{C}$  outputs a string  $\hat{\phi}$ . The adversary wins if either of the following conditions hold:

- $\phi = (k, \sigma) = (k, s, u)$ ,  $VERIFY(k, s, u) = \text{accept}$  and  $\mathcal{A}$  has not made *KRELEASE* query for  $s$  or,
- $\hat{\phi} = (k, \omega) = (k, s, v)$ ,  $VERIFY(k, s, v) = \text{accept}$ , and  $s = KTRAN(\hat{s}, x)$ ;  
 $\hat{\phi} = (k, \sigma) = (k, \hat{s}, \hat{u})$  and  $VERIFY(k, \hat{s}, \hat{u}) \neq \text{accept}$ .

**Definition 4.** A concurrent signature protocol is fair if the advantage of an polynomial-bounded adversary in the above game is negligible.

#### 4.3 Anonymity

**Definition 5.** A concurrent signature is anonymous if the following conditions hold:

- The value  $\sigma_i = \langle s, u_i \rangle = ISIGN(X_i, x_i, M_i)$  is indistinguishable from random elements of the signature space.
- The tuple  $(s, \omega_j)$  where  $s$  is the keystone fix input to *SSIGN* and  $\omega_j = \langle s', v_j \rangle = SSIGN(X_j, x_j, M_j, s)$  is indistinguishable from random elements of the signature space.

#### 4.4 Unlinkability

The unlinkability property is defined by the following game between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ .

*GAME 2:*

**Setup:** as SETUP in section 3.1. The challenger  $\mathcal{C}$  is also given two pairs of public and private keys  $(X, x)$  and  $(X', x')$ .

**Challenge:** The challenger simulates by himself two instances of the concurrent signature protocol (using his two different pairs of public and secret keys  $(X, x)$  and  $(X', x')$  to obtain two different pairs of converted signatures  $(\rho_0, \rho'_0)$  and  $(\rho_1, \rho'_1)$ . Here  $\rho_0$  and  $\rho_1$  are signatures of the same type (Schnorr or Schnorr-like) issued with the secret key  $x$ ; and  $\rho'_0$  and  $\rho'_1$  are signatures of the same type (Schnorr-like or Schnorr respectively) issued with the secret key  $x'$ . The challenger then chooses a random bit  $b \in \{0, 1\}$  and set  $\rho' = \rho'_b$ . The signatures  $\rho_0, \rho_1$  and  $\rho'$  are then given to the adversary along with the two private keys  $x$  and  $x'$ .

**Output:** Finally  $\mathcal{A}$  outputs the bit  $b'$ . The adversary wins the game if  $b = b'$ .

**Definition 6.** *A concurrent signature scheme is unlinkable if the probability of an adversary to win the above game is not better than  $1/2$ .*

This definition captures the intuition that if signatures are unlinkable, the probability that the adversary to identify the correct pair  $(\rho_b, \rho'_b)$  must not be non-negligibly better than a random toss of the coin. By letting the adversary to have the secret keys after the challenge is created, we essentially allow the adversary to have the complete view of all protocol runs thereafter. Here we made an implicit assumption that the messages signed are not linked.

## 5 An Asymmetric Concurrent Signature Scheme

We now proceed to give a concrete asymmetric concurrent signature scheme.

**Setup:** the public parameters are the primes  $p$  and  $q$  of appropriate size such that  $q|(p-1)$ , and a generator  $g$  for the subgroup in  $\mathbb{Z}_p^*$  of order  $q$ . The space  $\mathcal{M}$  is the set of all binary strings  $\{0, 1\}^*$ ,  $K = \mathbb{Z}_q$  and  $F$  is the subgroup of  $\mathbb{Z}_p^*$  generated by  $g$ . Private keys  $x_i$ 's are chosen randomly from  $\mathbb{Z}_q$ . Their corresponding public keys are  $X_i$ 's, each satisfies  $X_i = g^{x_i}$ .  $KGEN$  is defined as  $KGEN(x) = g^x$ , each  $KGEN_j$  is defined as  $KGEN_j(k) = X_j^k$ , and  $KTRAN$  is defined as  $KTRAN(s, x_j) = s^{x_j}$ .

**ISIGN:** on input of  $\langle X_i, x_i, M_i \rangle$ , the algorithm generates a random  $r \in \mathbb{Z}_q$  and returns the Schnorr promise of signature  $\sigma_i = \langle s, c \rangle$  where  $c = H(g^r, M_i)$ ,  $s = g^{r+cx_i}$ .

**SSIGN:** on input of  $\langle X_j, x_j, M_j \rangle$  and  $s$ , the algorithm generates a random  $r' \in \mathbb{Z}_q$  and returns the promise of Schnorr-like signature  $\omega_j = \langle s', \kappa_1, c' \rangle$  where  $s' = s^{x_j}$ ,  $c' = H(g^{r'} s', M_j)$  and  $\kappa_1 = (r' - c')/x_j \bmod q$ .

**IVERIFY:** on input of  $\sigma_i = \langle s, c \rangle$ , outputs *accept* if  $c = H(sX_i^{-c}, M_i)$  and *reject* otherwise.

**SVERIFY:** on input of  $\omega_j = \langle s', \kappa_1, c' \rangle$ , outputs *accept* if  $c' = H(g^{c'} X_i^{\kappa_1} s', M_j)$  and *reject* otherwise.

**VERIFY:** on input of the keystone  $k$  and the promise of signature  $\sigma_i = \langle s, c \rangle$  (or  $k$  and the promise of signature  $\omega_j = \langle s', \kappa_1, c' \rangle$ ), the algorithm outputs *accept* if  $KGEN(k) = s$  and  $IVERIFY(\sigma_i) = \textit{accept}$  (or respectively  $KGEN_j(k) = s'$  and  $SVERIFY(\omega_j) = \textit{accept}$ ).

## 5.1 Security

The completeness of the above concurrent signature scheme is by inspection. It remains to show that our concurrent signature protocol is fair and anonymous and the converted signatures are unlinkable.

**Theorem 1.** *The concurrent signature protocol constructed from the above concurrent signature scheme is fair, provided that the Schnorr and Schnorr-like signature schemes are unforgeable.*

*Proof.* The proof is by contradiction. If the concurrent signature protocol is not fair, by definition it must violate one of these two conditions with non-negligible probability:

**Case 1:** the party  $B$  can obtain a signature  $(k, \sigma) = (k, s, c)$  such that  $(k, s, c)$  is accepted by  $VERIFY$  without getting the keystone  $k$  from  $A$ .

**Case 2:** the party  $A$  can obtain a signature  $(k, \omega) = (k, s', \kappa_1, c')$  such that  $(k, s', \kappa_1, c')$  is accepted by  $VERIFY$  while the output of party  $B$   $(k, \sigma) = (k, s, c)$  is not accepted by  $VERIFY$ , i.e.,  $VERIFY(k, s, c) \neq \textit{accept}$ .

We consider each of the two cases.

**Case 1.**  $B$  is able to obtain a valid signature  $(k, \sigma)$  such that  $(k, s, c)$  is accepted by  $VERIFY$  without getting  $k$  from  $A$ . This is equivalent to  $B$  getting a valid signature  $(k, \sigma)$  from the promise of signature  $\sigma$ . This implies a Schnorr signature forgery.

**Case 2.**  $A$  is able to obtain a valid signature  $(k, \omega)$ .  $A$  could either receive  $\omega$  from  $B$  or generate  $\omega$  by herself. If  $A$  does not receive  $\omega$  from  $B$ ,  $A$  must generate the whole tuple  $(k, \omega)$  by herself. This means that  $A$  could forge the Schnorr-like signature  $(k, \omega)$  which contradicts the unforgeability property of the basic signature scheme.

If  $A$  receives the promise of signature  $\omega = \langle s', \kappa_1, c' \rangle$ , it means  $B$  must have obtained a promise of signature  $\sigma = \langle s, c \rangle$  such that  $s' = s^{x_B}$ . Since  $(k, \omega)$  is a valid signature, we have  $s' = X_B^k$ . This means  $s = s'^{1/x_B} = g^k$ ; and thus  $(k, \sigma)$  is a valid signature which contradicts the original assumption.

**Theorem 2.** *In the random oracle model, the concurrent signature protocol constructed from the above concurrent signature scheme is anonymous under the Decisional Diffie-Hellman assumption.*

*Proof.* By definition, our signature protocol is anonymous if these conditions are both satisfied:

**Case 1:** the promise of signature  $\sigma_i = \langle s, c \rangle$  outputted by the algorithm *ISSIGN* on input  $(X_i, x_i, M_i)$  is indistinguishable from random elements of the signature space and,

**Case 2:** the tuple  $(s, \omega_j)$  where  $\omega_j = \langle s', \kappa_1, c' \rangle$  is the output of algorithm *SSIGN* on input  $(X_j, x_j, M_j, s)$  is indistinguishable from random elements of the signature space.

Case 1 comes straight from the anonymity property of the Schnorr promise of signature. For Case 2, we show that if there is an oracle which distinguishes  $(s, \omega_j)$  from random elements of the signature space, there is a machine which could use the oracle to solve the Decisional Diffie-Hellman problem in the random oracle model.

Our machine interacts with the oracle as follows:

**Input:** a tuple of  $\langle g, g^x, g^y, g^r \rangle$ .

**Operation 1:** The machine serves as the hash function for the oracle. The machine maintains a list of previous hash query definitions. If a hash query is defined, the machine returns the previous output. Otherwise, the machine returns a random output and adds the entry pair to the definition list.

**Operation 2:** The machine chooses random  $c$  and  $\kappa_1$  and sets  $s = g^y, X_j = g^x, s' = g^z$ , and  $\omega_j = \langle s', \kappa, c \rangle$ . The machine adds the pair of  $(c, g^c X_j^{\kappa_1} s)$  to the list of hash function definition. Finally the machine sends the pair  $(s, \omega_j)$  to the oracle. This operation could be initiated by the machine or by the oracle. We place no restriction on how this operation could be initiated.

**Output:** If the oracle outputs that one of  $\{(s, \omega_j)\}$  is a valid promise of signature, the machine outputs  $\langle g, g^x, g^y, g^r \rangle$  is a valid Diffie-Hellman tuple, i.e.,  $r = xy$ .

If  $\langle g, g^x, g^y, g^r \rangle$  is a Diffie-Hellman tuple then  $r = xy$  and thus  $(s', \omega_j)$  is a valid pair (with  $k = x$ ). If  $\langle g, g^x, g^y, g^r \rangle$  is not a Diffie-Hellman tuple then  $\log_g s' \neq x$  and thus  $(s', \omega_j)$  is not a valid pair. Thus our machine will give a correct answer to the Decisional Diffie-Hellman problem if the machine would terminate and the oracle could distinguish a valid  $(s, \omega_j)$  from an invalid one. The latter is by the assumption of the oracle. For the former, it remains to show that the machine terminates in a polynomially-bounded number of operations.

It is clear that Operation 1 always terminates. Operation 2 would not terminate only if the hash query is already defined for challenge  $g^c X_j^{\kappa_1} s$ , i.e., there is another value  $c' \neq c$  such that  $c' = H(g^c X_j^{\kappa_1} s)$  is already defined by the machine. With random  $c$  and  $\kappa_1$ ,  $g^c X_j^{\kappa_1} s$  is uniformly distributed and the probability that it appears in a polynomial-bounded list is negligible.

**Theorem 3.** *Assuming the messages are unrelated, when converted to become ordinary signatures, our concurrent signatures are unlinkable under the random oracle model.*

*Proof.* According to definition 4, to prove that our concurrent signatures are unlinkable, we need to show that the advantage of an adversary to identify the bit  $b \in \{0, 1\}$  in GAME 2 (section 4.4) over the random toss of the coin is negligible. Since the messages are irrelevant to our discussion, we ignore the messages in the proof whenever possible.

We shall provide the proof for the case where  $\rho_0$  and  $\rho_1$  are Schnorr signatures and  $\rho'_b$  is a Schnorr-like signature. The case where  $\rho_0$  and  $\rho_1$  are Schnorr-like signatures and  $\rho'_b$  is a Schnorr signature is similar and omitted.

Let  $\rho_0 = (k_0, c_0)$  and  $\rho_1 = (k_1, c_1)$  where  $c_i = H(X^{-c_i}g^{k_i})$  ( $i = 1, 2$ ), and let  $\rho'_b = (k'_b, c'_b)$  where  $c'_b = H(X^{k'_b}g^{c'_b})$ .

Let  $r_0 = k_0 - c_0x \bmod q$  and  $r'_0 = (k'_b - k_0)x_j + c'_b \bmod q$ . Since  $c_0$  and  $c'_b$  are outputs of the random oracle  $H()$ , the pair  $\langle r_0, r'_0 \rangle$  is uniformly distributed in  $\mathbb{Z}_q \times \mathbb{Z}_q$ . It is easy to verify that the concurrent signature protocol in which  $r_0$  is the random input to the algorithm  $ISIGN(X, x, M_0)$  and  $r'_0$  is the random input to the algorithm  $SSIGN(X', x', M'_b)$  is legitimate and returns the signatures  $\rho_0$  and  $\rho'_b$ .

Similarly, let  $r_1 = k_1 - c_1x \bmod q$  and  $r'_1 = (k'_b - k_1)x_j + c'_b \bmod q$ . Since  $c_1$  and  $c'_b$  are outputs of the random oracle  $H()$ , the pair  $\langle r_1, r'_1 \rangle$  is uniformly distributed in  $\mathbb{Z}_q \times \mathbb{Z}_q$ . It is likewise easy to verify that the concurrent signature protocol in which  $r_1$  is the random input to the algorithm  $ISIGN(X, x, M_1)$  and  $r'_1$  is the random input to the algorithm  $SSIGN(X', x', M'_b)$  is legitimate and returns the signatures  $\rho_1$  and  $\rho'_b$ .

Then the probability for the adversary to identify the bit  $b$  is equal to the probability for the adversary to identify whether the random pair  $\langle r_0, r'_0 \rangle$  or  $\langle r_1, r'_1 \rangle$  is the actual pair  $\langle r_b, r'_b \rangle$  used in the protocol which generates  $\rho_b$  and  $\rho'_b$ .

Since  $r_b$  and  $r'_b$  are uniformly and independently chosen random values from  $\mathbb{Z}_q$ , the pair  $\langle r_b, r'_b \rangle$  is uniformly distributed in  $\mathbb{Z}_q \times \mathbb{Z}_q$ . This means the probability that  $\langle r_b, r'_b \rangle = \langle r_0, r'_0 \rangle$  is equal to the probability that  $\langle r_b, r'_b \rangle = \langle r_1, r'_1 \rangle$ , i.e., the probability for the adversary to win GAME 2 is no better than a random toss of the coin.

## 5.2 Discussion

The benefit of anonymity that our concurrent signature scheme offers might turn out to be a disadvantage to the matching signer in certain cases. Due to the anonymity property, the matching signer might not be able to distinguish faked queries from genuine ones and hence is vulnerable against denial-of-service attacks. This problem could be overcome by forcing the initial signer to prove in zero-knowledge the knowledge of the homomorphic inverse. This is easily accomplished using the standard Schnorr identification protocol [9]. The trade-off is a reduction in the protocol performance. We note that this problem is not applicable to the original CKP scheme as due to the nature of ring signatures, the matching signer can always determine if a signature sent from the initial signer is genuine or faked.

The protocol could be extended to work with multiple matching signers. In this model, one initial signer is used as the hub connecting with all multi-

ple matching signers. Each matching signer is assumed to behave like the the matching signer in the two-party case. It is easy to see that in this scenario, the anonymity of each and every signer is protected. Each concurrent signature in this case is still of the same size. When the keystone is revealed, all obtained signatures still remain unlinkable. We note that the CKP construction would require extra information proportional to the size of the group embedded in each signature. This results in concurrent signatures of the size proportional to the size of the group.

## References

1. N. Asokan, V. Shoup, and M. Waidner, *Optimistic Fair Exchange of Digital Signatures* (Extended Abstract), In K. Nyberg (ed.), EUROCRYPT '98, Lecture Notes in Computer Science vol. 1403, Springer-Verlag, Berlin, 1998, pp. 591-606.
2. N. Asokan, V. Shoup, and M. Waidner, *Optimistic fair exchange of signatures*, In IEEE Journal on Selected Areas in Communication vol. 18(4), 2000, pp. 593-610.
3. L. Chen, C.J. Kudla and K.G. Paterson, *Concurrent Signatures*. In C. Cachin and J. Camenisch (eds.), EUROCRYPT 2004, Lecture Notes in Computer Science vol. 3027, Springer-Verlag, 2004, pp. 287-305.
4. L.C. Guillou and J.-J. Quisquater, *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory*, EUROCRYPT 1988, Lecture Notes in Computer Science, Springer-Verlag, 1989, pp. 123-128.
5. National Institute of Standards and Technology, NST FIPS PUB 186, *Digital Signature Standard*, U.S. Department of Commerce, May, 1994.
6. D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures*, Journal of Cryptology, Volume 13 - Number 3, Springer-Verlag, 2000, pp. 361-396.
7. R. Rivest, A. Shamir, and L.M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communicaions of the ACM, vol. 21, no. 2, Feb 1978, pp. 120-126.
8. R. L. Rivest, A. Shamir, and Y. Tauman, *How to Leak a Secret*, C. Boyd (Ed.), ASIACRYPT 2001, Lecture Notes in Computer Science vol. 2248, Springer-Verlag, 2001, pp. 552-565.
9. C.P. Schnorr, *Efficient signature generation by smart cards*, In Journal of Cryptology, vol. 4, no. 3, 1991, pp. 161-174.
10. W. Susilo, Y. Mu and F. Zhang, *Perfect Concurrent Signature Schemes*, ICICS 2004, Lecture Notes in Computer Science vol. 3269, Springer-Verlag, 2004, pp. 14-27.

# Generic Construction of (Identity-Based) Perfect Concurrent Signatures

Sherman S.M. Chow<sup>1</sup> and Willy Susilo<sup>2</sup>

<sup>1</sup> Department of Computer Science,  
Courant Institute of Mathematical Sciences,  
New York University, NY 10012, USA  
schow@cs.nyu.edu

<sup>2</sup> Center for Information Security Research,  
School of Information Technology and Computer Science,  
University of Wollongong, Wollongong 2522, Australia  
wsusilo@uow.edu.au

**Abstract.** The notion of concurrent signatures was recently introduced by Chen, Kudla and Paterson. In concurrent signature schemes, two entities can produce two signatures that are *not* binding, until an extra piece of information (namely the *keystone*) is released by one of the parties. Subsequently, it was noted that the concurrent signature scheme proposed in the seminal paper cannot provide perfect ambiguity. Then, the notion of *perfect* concurrent signatures was introduced. In this paper, we define the notion of *identity-based (or ID-based) perfect concurrent signature schemes*. We provide the *first* generic construction of (ID-based) perfect concurrent signature schemes from ring signature schemes. Using the proposed framework, we give two concrete ID-based perfect concurrent signature schemes based on two major paradigms of ID-based ring signature schemes. Security proofs are based on the random oracle model.

**Keywords:** Concurrent Signatures, Perfect Ambiguity, Fair-Exchange, Ring Signatures, Identity-based Signatures, Bilinear Pairing.

## 1 Introduction

Consider the situation where a customer Alice would like to make a purchase request of a physical item from a shop owner Bob. One of the ways to do the transaction is asking Alice to firstly sign a payment instruction to pay Bob the price of the item. Then, Bob agrees by signing a statement that he authorizes Alice to pick the item up from the store, which will be sent via an email or other means upon receiving Alice's signature. We would like to make sure that both parties (the customer and the shop owner in our case) get the other party's item, or no party gets the other party's item at the end of a transaction, that is, the principle of fair exchange. For purchase occurred in a face-to-face manner, people have a higher confidence in getting back the other party's item shortly after giving out his or her item to be exchanged. However, to achieve fair exchange over Internet, in which two parties are mutually distrustful, is not a trivial task.

Concurrent signature can help when the full power of fair exchange is not necessary [6]. A pair of concurrent signatures can be made binding at the same time, i.e. when Alice picks up the item from Bob's store. At this time, Alice's signature (i.e. payment instruction) will be binding and Bob's signature (to allow Alice to pick up the item) will also be binding concurrently.

Subsequently, [13] noted that the concurrent signature scheme proposed in [6] cannot provide perfect ambiguity if both signers are known to be trustworthy. With the aim to further anonymize the signatures before the signatures are made binding, the notion of *perfect* concurrent signatures was introduced.

## 1.1 Related Work

Fair exchange of signature is a fundamental research problem in cryptography. Fairness in exchanging signatures is normally achieved with the help of a trusted third party (TTP) (which is often offline [2]). There were some attempts where a fair exchange of signatures can be achieved with a "semi-trusted" TTP who can be called upon to handle disputes between signers [1, 9]. This type of fair exchange is also referred to as an optimistic fair exchange. The well-known open problem in fair exchange is the requirement of a dispute resolving TTP whose role cannot be replaced by a normal certification authority (CA).

In [12], the notion of *ring signatures* was formalized and an efficient scheme based on RSA was proposed. A ring signature scheme allows a signer who knows at least one piece of secret information (or a trapdoor) to produce a sequence of  $n$  random permutations and form them into a ring. This ambiguous signature can be used to convince any third party that one of the people in the group (who knows the trapdoor information) has authenticated the message on behalf of the group. The authentication provides *signer ambiguity*, in the sense that no one can identify who has actually signed the message. The ID-based version of ring signature schemes was introduced in [14]. After that, a number of ID-based ring signature schemes were proposed. A recent study [7] showed that these schemes can be classified into two major paradigms, namely, the conversation from non-ID-based ring signature and the extension from ID-based signature. Please refer to [7] for a more detailed review of ID-based ring signature schemes.

## 1.2 Our Contributions

We define the notion of ID-based perfect concurrent signatures, which is the strongest notion (in terms of privacy) of concurrent signature currently available. We provide a generic construction of both non-ID-based and ID-based perfect concurrent signature schemes from ring signatures, which is the first discussion in the literature. We illustrate our idea by two schemes from each of two major paradigms of existing ID-based ring signature schemes. Both of them enjoy short signature length which is only one group element on elliptic curve larger than most existing ID-based signature schemes, our second scheme is also efficient in the sense that no pairing operation is required for the generation of signature.



### 1.3 Paper Organization

The rest of this paper is organized as follows. The next section reviews some notions that will be used throughout this paper. Section 3 provides a model of ID-based perfect concurrent signature schemes together with its security requirements. We also present a generic construction of (ID-based) perfect concurrent signature protocol in this section. In Section 4 and Section 5, we provide two concrete ID-based perfect concurrent signature schemes. Section 6 concludes the paper and discusses future research direction.

## 2 Preliminaries

### 2.1 Basic Concepts on Bilinear Pairings

Let  $\mathbb{G}_1, \mathbb{G}_2$  be cyclic additive groups generated by  $P_1, P_2$ , respectively, whose order are a prime  $q$ . Let  $\mathbb{G}_M$  be a cyclic multiplicative group with the same order  $q$ . We assume there is an isomorphism  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$  such that  $\psi(P_2) = P_1$ . Let  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_M$  be a bilinear mapping with the following properties:

1. *Bilinearity*:  $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$  for all  $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b, \in \mathbb{Z}_q$ .
2. *Non-degeneracy*: There exists  $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$  such that  $\hat{e}(P, Q) \neq 1$ .
3. *Computability*: There exists an efficient algorithm to compute  $\hat{e}(P, Q)$  for all  $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ .

For simplicity, hereafter, we set  $\mathbb{G}_1 = \mathbb{G}_2$  and  $P_1 = P_2$ . We note that our scheme can be easily modified for a general case, when  $\mathbb{G}_1 \neq \mathbb{G}_2$ .

A bilinear pairing instance generator is defined as a probabilistic polynomial time algorithm  $\mathcal{IG}$  that takes as input a security parameter  $\ell$  and returns a uniformly random tuple  $param = (p, \mathbb{G}_1, \mathbb{G}_M, \hat{e}, P)$  of bilinear parameters, including a prime number  $p$  of size  $\ell$ , a cyclic additive group  $\mathbb{G}_1$  of order  $q$ , a multiplicative group  $\mathbb{G}_M$  of order  $q$ , a bilinear map  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_M$  and a generator  $P$  of  $\mathbb{G}_1$ . For a group  $\mathbb{G}$  of prime order, we denote the set  $\mathbb{G}^* = \mathbb{G} \setminus \{\mathcal{O}\}$  where  $\mathcal{O}$  is the identity element of the group.

### 2.2 Complexity Assumption

#### Definition 1. Computational Co-Diffie-Hellman (Co-CDH) Problem.

Given a randomly chosen  $(P_1, P_2, aP_1, bP_2)$ , where  $P_1, P_2 \in \mathbb{G}_1, a, b \in \mathbb{Z}_q^*$ , and  $a, b$  are unknown, compute  $abP_2 \in \mathbb{G}_M$ .

#### Definition 2. Co-CDH Assumption.

If  $\mathcal{IG}$  is a Co-CDH parameter generator, the advantage  $\text{Adv}_{\mathcal{IG}}(\mathcal{A})$  that an algorithm  $\mathcal{A}$  has in solving the Co-CDH problem is defined to be the probability that the algorithm  $\mathcal{A}$  outputs  $abP_2$  on inputs  $\mathbb{G}_1, \mathbb{G}_M, P_1, P_2, aP_1, bP_2$ , where  $(\mathbb{G}_1, \mathbb{G}_M)$  is the output of  $\mathcal{IG}$  for sufficiently large security parameter  $\ell$ ,  $P_1, P_2$  are random generators of  $\mathbb{G}_1$  and  $a, b$  are random elements of  $\mathbb{Z}_q^*$ . The Co-CDH assumption is that  $\text{Adv}_{\mathcal{IG}}(\mathcal{A})$  is negligible for all efficient algorithms  $\mathcal{A}$ .

### 2.3 Review on Concurrent Signatures

In concurrent signatures, there are two parties involved in the protocol, namely  $A$  and  $B$  (or Alice and Bob, respectively). At first, both parties' signatures are ambiguous from any third party's point of view, but they will be simultaneously binding *after* an additional information, called a “*keystone*” is released by one of the participants. Since one party is required to create a keystone and send the first message to the other party, we call this party the *initial signer*. A party who responds to the initial signature by creating another signature is called a *matching signer*. We note that if Alice does not release the keystone, then the transaction cannot be completed, although Bob would like to do so. Nevertheless, there are many scenarios where this type of signature schemes is applicable [6].

Similar to the definition in [6], concurrent signatures are digital signature schemes that consist of the following algorithms:

- **SETUP**: A probabilistic algorithm that accepts a security parameter  $\ell$ , outputs the descriptions of the message space  $\mathcal{M}$ , the signature space  $\mathcal{S}$ , the keystone space  $\mathcal{K}$ , the keystone fix space  $\mathcal{F}$ , a function  $KSGEN : \mathcal{K} \rightarrow \mathcal{F}$  and any other parameters  $\pi$ .
- **KEYGEN**: A probabilistic algorithm that accepts a security parameter  $\ell$ , outputs the public key  $y_i$ ; together with the corresponding private key  $x_i$  to be kept secretly.
- **ASIGN**: A probabilistic algorithm that accepts  $(y_i, y_j, x_i, h_1, h_2, m)$ , where  $h_1, h_2 \in \mathcal{F}$ ,  $y_i, y_j \neq y_i$  are public keys,  $x_i$  is the private key corresponding to  $y_i$ , and  $m \in \mathcal{M}$ , outputs a *signer-ambiguous* signature  $\sigma = (s, h_1, h_2)$  where  $s \in \mathcal{S}$ , and  $h_1, h_2$  are the sources of randomness used in generating  $s$ .
- **AVERIFY**: An algorithm that accepts  $S = (\sigma, y_i, y_j, m)$ , where  $\sigma = (s, h_1, h_2)$ ,  $s \in \mathcal{S}$ ,  $h_1, h_2 \in \mathcal{F}$ ,  $y_i$  and  $y_j$  are public keys, and  $m \in \mathcal{M}$ , outputs **accept** or **reject**. The symmetric property of **AVERIFY** requires  $\text{AVERIFY}(\sigma', y_j, y_i, m) = \text{AVERIFY}(\sigma, y_i, y_j, m)$  for  $\sigma' = (s, h_2, h_1)$ .
- **VERIFY**: An algorithm that accepts  $(k, S)$  where  $k \in \mathcal{K}$  is a keystone and  $S$  is of the form  $S = (\sigma, y_i, y_j, m)$ , where  $\sigma = (s, h_1, h_2)$  with  $s \in \mathcal{S}$ ,  $h_1, h_2 \in \mathcal{F}$ ,  $y_i$  and  $y_j$  are public keys, and  $m \in \mathcal{M}$ . The algorithm verifies whether  $KSGEN(k) \stackrel{?}{=} h_2$  holds. If it does not hold, then it terminates with output **reject**. Otherwise, it runs **AVERIFY**( $S$ ).

As discussed in the introduction, the concrete construction of concurrent signature schemes in [6] cannot provide *perfect* ambiguity in certain situations. In their scheme, the two signatures have an explicit relationship which can be easily observable by any third party. As a consequence, when the two signers are well known to be honest that will always conform to the protocol, then any third party would trust that the signatures are valid. Since the signatures can be identified even *before* the keystone is released, it contradicts with the requirement of concurrent signatures. Concurrent signature schemes with perfect ambiguity was considered in [13]. They presented two schemes based on the discrete logarithm problem and bilinear pairings. Their constructions are based on the framework proposed by [6], and they have not considered the generic construction of perfect concurrent signature schemes.

### 3 Generic Framework and Security Notions

We note that the algorithms listed out by [6] may not be enough to cater for the need of perfect ambiguity. In view of this, we provide a new generic framework.

#### 3.1 Building Blocks

Firstly, we provide a formal definition of the algorithm used in our generic construction of perfect concurrent signature schemes, by incorporating some elements from the notion introduced in [6]. Notice that to achieve the perfect ambiguity, we no longer require the matching signer to use the same keystone fix as the initial signer. Beside, a pair of keystones is used instead of a single one. We also describe the essential properties of these algorithms for the construction of perfect concurrent signature schemes.

**Definition 3.** *A perfect concurrent signature scheme is a digital signature scheme that consists of the following algorithms:*

- **SETUP:** *A probabilistic algorithm that on input a security parameter  $\ell$ , outputs the system parameters  $\text{params}$  which is the descriptions of the the message space  $\mathcal{M}$ , the signature space  $\mathcal{S}$ , the keystone-pair space  $\mathcal{K}_I \times \mathcal{K}_M$ , the keystone fix space  $\mathcal{F}$  and the encrypted keystone space  $\mathcal{K}'$ . Note that we do not include  $\text{params}$  explicitly as the input in the following descriptions.*
- **KEYGEN:** *A probabilistic algorithm that is invoked by a participant  $\text{ID}$ . The algorithm outputs a public key  $\text{Q}_{\text{ID}}$  and the corresponding the secret key  $\text{S}_{\text{ID}}$ .*
- **FIX-INITIAL-KEYSTONE:** *A deterministic algorithm that on input a initial-keystone  $k_I \in \mathcal{K}_I$ , it outputs the corresponding keystone fix  $f_I \in \mathcal{F}$ .*
- **ASIGN:** *A probabilistic algorithm that on inputs  $(\text{ID}_i, \text{ID}_j, \text{S}_{\text{ID}_i}, \alpha, f, m)$ , where  $\alpha, f \in \mathcal{F}$ ,  $\text{ID}_i, \text{ID}_j$  are the identities of the participants,  $\text{S}_{\text{ID}_i}$  is the secret key associated with  $\text{ID}_i$ , and  $m \in \mathcal{M}$ , outputs an ambiguous signature  $\sigma = \{U_i, U_j, V\}$  on  $m$ .*
- **ENC-MATCHING-KEYSTONE:** *A deterministic algorithm that on input a matching-keystone  $k_M \in \mathcal{K}_M$ , it outputs the encrypted matching-keystone  $K_M \in \mathcal{K}'$ .*
- **FIX-SECRET-KEYSTONE:** *A deterministic algorithm that on inputs an encrypted matching-keystone  $K_M \in \mathcal{K}'$  and a secret key  $\text{S}_{\text{ID}_i}$ , outputs a secret keystone fix  $f_S \in \mathcal{F}$ .*
- **AVERIFY:** *A deterministic algorithm that takes as input  $S = (\sigma, \text{ID}_i, \text{ID}_j, m)$  and outputs **accept** or **reject**. Again we require a symmetric property that  $\text{AVERIFY}(\sigma, \text{ID}_i, \text{ID}_j, m) = \text{AVERIFY}(\sigma', \text{ID}_j, \text{ID}_i, m)$  for  $\sigma' = \{U_j, U_i, V\}$ .*
- **VERIFY-INITIAL-KEYSTONE:** *A deterministic algorithm that on input an initial-keystone  $k_I \in \mathcal{K}_I$  and its corresponding fix  $k_I \in \mathcal{F}$ , it outputs **accept** or **reject** by checking  $f_I \stackrel{?}{=} \text{FIX-INITIAL-KEYSTONE}(k_I)$ .*
- **VERIFY-SECRET-KEYSTONE:** *A deterministic algorithm that on input a matching-keystone  $k_M \in \mathcal{K}_M$ , a secret keystone fix  $f_S \in \mathcal{F}$  and an identity  $\text{ID}_i$ , it outputs **accept** or **reject** depending whether  $f_S = \text{FIX-SECRET-KEYSTONE}(\text{ENC-MATCHING-KEYSTONE}(k_M), \text{S}_{\text{ID}_i})$ .*

- **VERIFY-CONNECTION**: A deterministic algorithm that on input a pair of signatures  $\sigma_i = \{U_i, U_j, V\}$  and  $\sigma_j = \{U'_i, U'_j, V'\}$  and a pair of keystone fix  $f_I$  and  $f_S$ , it outputs **accept** or **reject** depending whether  $U_j = f_I$  and  $U'_i = U_j \otimes f_S$ , where  $\otimes$  is the operator of the group  $\mathcal{F}$ .
- **VERIFY**: A deterministic algorithm that takes as input  $(k_I, k_M, S')$ , where  $(k_I, k_M) \in \mathcal{K}_I \times \mathcal{K}_M$ ,  $S' = (\sigma_i, \sigma_j, \text{ID}_i, \text{ID}_j, m)$ . The algorithm verifies if all of **VERIFY-INITIAL-STONE**, **VERIFY-SECRET-KEYSTONE** and **VERIFY-CONNECTION** are true. If not, it terminates with output **reject**. Otherwise, it runs **AVERIFY**( $S$ ) and the output of this algorithm is the output of the **AVERIFY** algorithm.

### 3.2 ID-Based Scenario

For ID-based perfect concurrent signature, we need to modify the **SETUP** algorithm described and replace **KEYGEN** algorithm by a new **EXTRACT** algorithm in the above definition.

**Definition 4.** An ID-based perfect concurrent signature scheme requires the following algorithms:

- **SETUP**: A probabilistic algorithm that on input a security parameter  $\ell$ , outputs descriptions of the set of participants  $\mathcal{U}$ , the message space  $\mathcal{M}$ , the signature space  $\mathcal{S}$ , the keystone-pair space  $\mathcal{K}_I \times \mathcal{K}_M$ , the keystone fix space  $\mathcal{F}$ , and the encrypted keystone space  $\mathcal{K}'$ . The algorithm also outputs the public key of the private key generator (**PKG**) and the master secret key of the **PKG** for the extraction of user’s private key.
- **EXTRACT**: A deterministic algorithm that is invoked by a participant and the **PKG**. On input an **ID** of a participant, the algorithm outputs a participant’s secret key  $\mathcal{S}_{\text{ID}}$ .

### 3.3 Generic Construction

In this section, we describe a generic construction of (ID-based) concurrent signature protocol. We highlight the properties of the algorithm involved. There are two parties, namely  $A$  (Alice) and  $B$  (Bob) that are involved in the protocol. Without losing generality, we assume that  $A$  is the initial signer and  $B$  is the matching signer. The protocol works as follows.

Firstly, **CA/PKG** runs the **SETUP** algorithm to determine the public parameters of the scheme. Then, depending on whether the scheme is ID-based, user invokes the corresponding algorithm to get the public-private key pair.

More specifically, for non-ID-based scenario, both  $A$  and  $B$  run **KEYGEN** to generate a public-private key pair (denoted by  $(Q_{\text{ID}_A}, S_{\text{ID}_A})$  and  $(Q_{\text{ID}_B}, S_{\text{ID}_B})$  respectively), register the public key and the identity with the **CA**, and possibly provides a proof-of-knowledge of the private key to the **CA** as well. After authentication (and the checking of the proof-of-knowledge) the **CA** issues a digital certificate binding the relation of the purported identity to the user.

For the ID-based scenario, both  $A$  and  $B$  visit the PKG and engage in the EXTRACT algorithm to obtain their secret key  $\mathcal{S}_{\text{ID}_A}$  and  $\mathcal{S}_{\text{ID}_B}$ , respectively. The identities of  $A$  and  $B$  are available publicly as  $\text{ID}_A$  and  $\text{ID}_B$ , together with public hash functions  $H_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ . Hence, the public key  $\text{Q}_{\text{ID}_i}$  can be computed by anyone (for instance, by computing  $\text{Q}_{\text{ID}_i} = H_0(\text{ID}_i)$ ).

After both users got their corresponding key pair, the protocol is as follows.

1.  $A$  picks a random keystone  $(k_I, k_M) \in K_I \times K_M$  and executes the SET-INITIAL-KEYSTONE algorithm using  $k_I$  as the input to obtain  $f_I \in \mathcal{F}$ .

A good candidate for SET-INITIAL-KEYSTONE is a cryptographic hash function, since it is hard to invert, i.e. given  $y$  from the range of the function, it is hard to find the pre-image  $x$ .

2.  $A$  selects a message  $m_A \in \mathcal{M}$ , together with her identity  $\text{ID}_A$  and  $B$ 's identity  $\text{ID}_B$ , and computes her ambiguous signature as  $\sigma_A = \{U_A, U_B, V\} \leftarrow \text{ASIGN}(\text{ID}_A, \text{ID}_B, \mathcal{S}_{\text{ID}_A}, \mathcal{O}_{\mathcal{F}}, f_I, m_A)$  where  $\mathcal{O}_{\mathcal{F}}$  denotes the identity element of the group  $\mathcal{F}$ . ( $\mathcal{O}_{\mathcal{F}}$  is used to unify the list of input parameters used by  $\text{ID}_A$  and  $\text{ID}_B$  for the ASIGN algorithm, which merely means that  $A$  can skip a certain group operation inside the ASIGN algorithm that is used to connect  $B$ 's signature with  $A$ 's.)

We require that the ASIGN algorithm to be able to produce ambiguous signature  $\sigma$  such that any one can get convinced that either  $\mathcal{S}_{\text{ID}_A}$  or  $\mathcal{S}_{\text{ID}_B}$  is used as the input but does not know exactly which one with probability greater than  $1/2$ . Moreover, there are two parts (which can be implicitly) involved with the signature such that the first part can be chosen arbitrary while the value of another part must be depending on the first part. Most of existing ring signature schemes satisfy these properties.

3.  $A$  hides the second keystone  $k_M$  by executing the ENC-MATCHING-KEYSTONE algorithm using  $k_M$  as the input to obtain  $K_M \in \mathcal{K}'$ ,  $A$  then sends  $K_M$  and  $\sigma_A$  to  $B$ . We require that  $k_M$  cannot be effectively computable from  $K_M$ . The choice of implementation for ENC-MATCHING-KEYSTONE will be discussed shortly afterward.

4. Upon receiving  $A$ 's ambiguous signature  $\sigma_A$ ,  $B$  verifies the signature by testing whether  $\text{AVERIFY}(\sigma_A, \text{ID}_A, \text{ID}_B, m_A) \stackrel{?}{=} \text{accept}$  holds. Obviously, the AVERIFY algorithm is simply the one matching with the ASIGN algorithm.

5.  $B$  aborts if the above equation does not hold. Otherwise,  $B$  picks a message  $m_B \in \mathcal{M}$  to sign.  $B$  firstly executes FIX-SECRET-KEYSTONE with the encrypted matching keystone  $K_M$  and his secret key  $\mathcal{S}_{\text{ID}_B}$  as input to obtain a secret matching keystone  $f_S$ , then computes his ambiguous message  $\sigma_B = \{U'_A, U'_B, V'\} \leftarrow \text{ASIGN}(\text{ID}_B, \text{ID}_A, \mathcal{S}_{\text{ID}_B}, U_B, f_S, m_B)$  and sends this value to  $A$ . We require that the value of  $f_S$  is uniquely determined by  $k_M$  and  $\mathcal{S}_{\text{ID}_B}$  and cannot be computed without the knowledge of  $\mathcal{S}_{\text{ID}_B}$  or  $k_M$  (that is why the keystone-fix is called a secret), yet its correctness can be checked by only knowing  $k_M$ . All these properties can be achieved by probabilistic public key encryption, such that  $k_M$  is the randomness used in encryption,  $K_M$  is part of the ciphertext, which can be viewed as a kind of

session key employed by probabilistic public key encryption. The value of  $f_S$  can be verified by using the knowledge of  $k_M$  and the recipient's public key  $Q_{ID_B}$ .

6. Upon receiving  $B$ 's ambiguous signature  $\sigma_B$ ,  $A$  verifies it by testing whether
- $\text{VERIFY-SECRET-KEYSTONE}(k_M, f_S, Q_{ID_B}) \stackrel{?}{=} \text{accept}$ ,
  - $\text{VERIFY-CONNECTION}(f_I, f_S, \sigma_A, \sigma_B) \stackrel{?}{=} \text{accept}$  and
  - $\text{AVERIFY}(\sigma_B, ID_B, ID_A, m_B) \stackrel{?}{=} \text{accept}$
- all hold. If not, then  $A$  aborts. Otherwise,  $A$  releases the keystone  $(k_I, k_M)$  to  $B$ , and both signatures are binding concurrently.

### 3.4 Security Notions

As the original model of concurrent signatures in [6], we require a perfect concurrent signatures (either ID-based or not) to satisfy *correctness*, *ambiguity*, *unforgeability* and *fairness*. Intuitively, these notions are described as follows. Note that we follow the definition of ambiguity in [13] instead of the one in [6].

- *Correctness*: If a signature  $\sigma$  has been generated *correctly* by invoking ASIGN algorithm on a message  $m \in \mathcal{M}$ , AVERIFY algorithm will return “accept” with an overwhelming probability, given a signature  $\sigma$  on  $m$  and a security parameter  $\ell$ . Moreover, after the keystone-pair  $(k_I, k_M) \in \mathcal{K}_I \times \mathcal{K}_M$ , is released, then the output of VERIFY algorithm will be “accept” with an overwhelming probability.
- *Ambiguity*: We require that given the two ambiguous signatures  $(\sigma_1, \sigma_2)$ , any adversary will not be able to distinguish who was the actual signer of the signatures *before* the keystone is released. Any adversary can only conclude that one of the following events has occurred:
  1. Both  $\sigma_1$  and  $\sigma_2$  were generated by the initial signer.
  2. Both  $\sigma_1$  and  $\sigma_2$  were generated by the matching signer.
  3. The initial signer generated  $\sigma_1$  while the matching signer generated  $\sigma_2$ .
  4. The matching signer generated  $\sigma_1$  while the initial signer generated  $\sigma_2$ .
 All these cases are equally probable from the adversary's view.
- *Unforgeability*: There are two levels of unforgeability to be considered.
  - *Level 1*: When an adversary  $\mathcal{A}$  does not have any knowledge of the respective secret key  $S_{ID}$ , then no valid signature that will pass the AVERIFY algorithm can be produced. Otherwise, one of the underlying hard problems can be solved by using this adversary's capability. This requirement is for the matching signer to get convinced that the signature presented by the initial signer is indeed originated from her.
  - *Level 2*: Any party cannot *frame* the other party that he or she has indeed signed a message. We require that although both signatures are ambiguous, any party who would like to frame (or cheat) the others will not be able to produce a valid keystone with an overwhelming probability. This means that the first signature can only be generated by the initial signer and it is unforgeable by anyone else, including the matching

signer. At the same time, the second signature can only originate from the matching signer, which is unforgeable by any person other than him, including the initial signer.

- *Fairness*: We require that any valid ambiguous signatures generated using the same keystone will all become binding *after* the keystone is released. Hence, a matching signer cannot be left in a position where a keystone binds his signature to him whilst the initial signer’s signature is not binding to her. This requirement is important for the case like the initial signer try to present a signature of another message after the matching signer has verified the validity of the original message and complete his part of protocol. However, we do not require that the matching signer will definitely receive the necessary keystone.

**Definition 5.** *An ID-based perfect concurrent signature scheme is secure if it is existentially unforgeable under a chosen message attack, ambiguous and fair.*

## 4 A Concrete Instantiation

We present a concrete ID-based perfect concurrent signature scheme using the above general construction, with the ID-based ring signature scheme proposed by Zhang and Kim [14] and the basic version of ID-based encryption scheme with semantic security proposed by Boneh and Franklin [4]. Using our generic construction in Section 3, we define the required eleven algorithms.

- **SETUP**: The PKG selects a random number  $s \in \mathbb{Z}_q^*$  and sets  $P_{pub} = sP$ . It selects three cryptographic hash functions  $H_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1$  and  $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ . and  $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ . It publishes system parameters  $params = \{\mathbb{G}_1, \mathbb{G}_M, \hat{e}, q, P, P_{pub}, H_0, H_1, H_2\}$ , and keeps  $s$  as the *master secret key*. The algorithm also sets  $\mathcal{M} = \mathcal{K}_I = \mathcal{K}_M = \mathcal{F} = \mathbb{Z}_q$  and  $\mathcal{K}' = \mathbb{G}_1$ .
- **EXTRACT**: The EXTRACT algorithm is defined as follows.
  1. A user  $\mathcal{U}_i$  submits his or her identity  $ID_i$  to the PKG.
  2. After a successful identification, PKG generates  $\mathcal{U}_i$  secret key as follows.
    - Compute  $Q_{ID_i} = H_0(ID_i)$ .
    - Compute  $\mathcal{U}_i$ ’s secret key as  $S_{ID_i} = sQ_{ID_i}$ .
    - Deliver  $S_{ID_i}$  as user  $\mathcal{U}_i$ ’s secret key through a private and authenticated channel.
- **FIX-INITIAL-KEYSTONE**: Assuming a keystone  $k_I \in \mathbb{Z}_q$  is randomly selected, this algorithm outputs  $f_I = H_1(k_I)$  as the keystone-fix.
- **ASIGN**: The ASIGN algorithm accepts the following parameters  $(ID_i, ID_j, S_{ID_i}, \alpha, f, m)$ , where  $S_{ID_i}$  is the secret key associated with  $Q_{ID_i}$ ,  $f \in \mathcal{F}$  and  $m \in \mathcal{M}$  is the message. The algorithm will perform the following.
  1. Select a random point  $Z \in \mathbb{G}_1^*$ .
  2. Set  $u_j \leftarrow \alpha \cdot f$ .
  3. Compute  $u_0 = H_1(H_2(m) || (ID_i \oplus ID_j) || \hat{e}(Z, P) \hat{e}(u_j Q_{ID_j}, P_{pub}))$ .
  4. Compute  $V = u_0^{-1}(Z - (u_0 - u_j)S_{ID_i})$ .
  5. Output  $\sigma = (u_i = u_0 - u_j, u_j, V)$  as the signature on message  $m$ .

- ENC-MATCHING-KEYSTONE: Assuming a keystone  $k_M \in \mathbb{Z}_q$  is randomly selected, this algorithm outputs  $K_M = k_M P$  as the encrypted keystone.
- FIX-SECRET-KEYSTONE: This algorithm returns  $H_1(\hat{e}(K_M, \mathcal{S}_{ID_j}))$ .
- AVERIFY. The AVERIFY algorithm accepts  $(\sigma, ID_i, ID_j, m)$ , where  $\sigma = (u_i, u_j, V)$ . The algorithm verifies whether

$$u_i + u_j \stackrel{?}{=} H_1 \left( H_2(m) \parallel (ID_i \oplus ID_j) \parallel \hat{e}(V, P)^{(u_i+u_j)} \hat{e}(u_i Q_{ID_i}, P_{pub}) \hat{e}(u_j Q_{ID_j}, P_{pub}) \right)$$

holds with equality. If so, then output **accept**. Otherwise, output **reject**.

- VERIFY-INITIAL-KEYSTONE: This algorithm outputs **accept** if  $f_I = H_1(k_I)$ , **reject** otherwise.
- VERIFY-SECRET-KEYSTONE: It outputs **accept** if  $f_S = H_1(\hat{e}(P_{pub}, Q_{ID_j})^{k_M})$ , **reject** otherwise.
- VERIFY-CONNECTION: This algorithm outputs **accept** if  $U'_i = U_j \cdot f_S$ , **reject** otherwise.
- VERIFY. The algorithm accepts  $(k_I, k_M, S')$ , where  $k_I \in \mathcal{K}_I$  and  $k_M \in \mathcal{K}_M$  are the keystones and  $S' = (m, \sigma_i, \sigma_j, ID_i, ID_j)$ , where  $\sigma = (u_i, u_j, V)$ . The algorithm verifies whether  $(k_I, k_M)$  is valid, by using the above two algorithm. If it does not hold, then output **reject**. Otherwise, run AVERIFY( $S$ ). The output of VERIFY is the output of AVERIFY algorithm.

*Correctness.*

The correctness of the above proposed scheme is justified as follows.

$$\begin{aligned} u_i + u_j &= H_1 \left( H_2(m) \parallel (ID_i \oplus ID_j) \parallel \hat{e}(V, P)^{(u_i+u_j)} \hat{e}(u_i Q_{ID_i}, P_{pub}) \hat{e}(u_j Q_{ID_j}, P_{pub}) \right) \\ u_0 &= H_1 \left( H_2(m) \parallel (ID_i \oplus ID_j) \parallel \hat{e}((u_i + u_j)V + u_i \mathcal{S}_{ID_i}, P) \hat{e}(u_j Q_{ID_j}, P_{pub}) \right) \\ &= H_1 \left( H_2(m) \parallel (ID_i \oplus ID_j) \parallel \hat{e}(u_0 V + (u_0 - u_j) \mathcal{S}_{ID_i}, P) \hat{e}(u_j Q_{ID_j}, P_{pub}) \right) \\ &= H_1 \left( H_2(m) \parallel (ID_i \oplus ID_j) \parallel \hat{e}(Z, P) \hat{e}(u_j Q_{ID_j}, P_{pub}) \right) \quad \blacksquare \end{aligned}$$

**4.1 Security Consideration**

The security proofs are omitted due to space limitation. We refer the reader to the full version of this paper for a more complex account.

**Theorem 1. (Ambiguity)** *Before the keystone  $k$  is released, both signatures are ambiguous.*

**Lemma 1.** *When the output of VERIFY is **accept**, then any third party can be sure who has generated the signature. Any party cannot frame that the other party has signed a message without his or her consent assuming the one-way property of the hash function. This guarantees that the signature is unforgeable.*

**Theorem 2. (Unforgeability)** *The scheme presented in this section is existentially unforgeable under a chosen message attack in the random oracle model, assuming the one-way property of the hash function, the hardness of the discrete logarithm problem and the Co-CDH assumption.*



**Theorem 3. (Fairness)** *For all signatures that are generated with the same keystone will be binding concurrently when the keystone is released.*

**Theorem 4.** *Our ID-based perfect concurrent signature scheme presented in this scheme is secure in the random oracle model, assuming the hardness of the discrete logarithm problem.*

### 4.2 Signature Length

In the above scheme, each signature is a three-tuple  $\sigma_i = (u_1, u_2, V)$ , where  $u_1, u_2 \in \mathbb{Z}_q$  and  $V \in \mathbb{G}_1$ . Using any of the families of curves described in [5], one can take  $q$  to be a 170-bit prime and use a group  $\mathbb{G}_1$  where each element is 171 bits. For example,  $\mathbb{G}_1$  is derived from the curve  $E/GF(3^{97})$  defined by  $y^2 = x^3 - x + 1$ , which has 923-bit discrete-logarithm security. With these choices, the total signature length for a pair of signature is 1,022 bits or 128 bytes.

## 5 A More Efficient Construction

Now we present a more efficient variant of ID-based perfect concurrent signature which requires no pairing operation in signing without sacrificing the computational efficiency of verification or other steps. Again, the construction follows our idea of generic construction in Section 3. We utilize the ID-based ring signature scheme proposed by Chow *et al.* [8] and also the basic version of ID-based encryption scheme with semantic security proposed in [4].

- **SETUP:** Basically it is the same as our first scheme, but the description of spaces become  $\mathcal{M} = \mathcal{K}_I = \mathcal{K}_M = \mathbb{Z}_q$ ,  $\mathcal{F} = \mathcal{K}' = \mathbb{G}_1$ .
- **EXTRACT:** The same as our first scheme.
- **FIX-INITIAL-KEYSTONE:** Assuming a keystone  $k_I \in \mathbb{G}_2$  is randomly selected, this algorithm outputs  $f_I = H_2(k_I)$  as the keystone-fix.
- **ASIGN:** The input of this algorithm includes two identities  $ID_i$  and  $ID_j$ , a private key  $\mathcal{S}_{ID_i}$ , a message  $m$ , a  $\mathbb{G}_1$  element  $\alpha$ , and a  $\mathbb{G}_1$  element  $f$ .
  1. Compute  $U_j = \alpha + f$  and  $h_j = H_1(m || (ID_i \oplus ID_j) || U_j)$ .
  2. Choose  $r'_i \in_R \mathbb{Z}_q^*$ , compute  $U_i = r'_i Q_{ID_i} - U_j - h_j Q_{ID_j}$ .
  3. Compute  $h_i = H_1(m || (ID_i \oplus ID_j) || U_i)$  and  $V = (h_i + r'_i) \mathcal{S}_{ID_i}$ .
  4. Output the signature  $\sigma = \{U_i, U_j, V\}$ .
- **ENC-MATCHING-KEYSTONE:** The same as our first scheme.
- **FIX-SECRET-KEYSTONE:** This algorithm returns  $f_S = H_2(\hat{e}(K_M, \mathcal{S}_{ID_j}))$ .
- **AVERIFY:** The input of this algorithm includes two identities  $ID_i$  and  $ID_j$ , a message  $m$ , and a ring signature  $\sigma = \{U_i, U_j, V\}$ .
  1. Compute  $h_i = H_1(m || (ID_i \oplus ID_j) || U_i)$  and  $h_j = H_1(m || (ID_i \oplus ID_j) || U_j)$ .
  2. Return **accept** if  $\hat{e}(P_{pub}, U_i + h_i Q_{ID_i} + U_j + h_j Q_{ID_j}) = \hat{e}(P, V)$ , **reject** otherwise.
- **VERIFY-INITIAL-KEYSTONE:** This algorithm outputs **accept** if  $f_I = H_2(k_I)$ , **reject** otherwise.

- VERIFY-SECRET-KEYSTONE: It outputs **accept** if  $f_S = H_2(\hat{e}(P_{pub}, Q_{ID_j})^{k_M})$ , **reject** otherwise.
- VERIFY-CONNECTION: This algorithm outputs **accept** if  $U'_i = U_j + f_S$ , **reject** otherwise.
- VERIFY. The algorithm accepts  $(k_I, m_I, S')$ , where  $k_I \in \mathcal{K}_I$  and  $k_M \in \mathcal{K}_M$  are the keystones and  $S' = (m, \sigma_i, \sigma_j, ID_i, ID_j)$ , where  $\sigma = (U_i, U_j, V)$ . The algorithm verifies whether  $(k_I, k_M)$  is valid and the connection between  $\sigma_i$  and  $\sigma_j$  is valid by using the above three algorithm. If it does not hold, then output **reject**. Otherwise, run AVERIFY( $S$ ). The output of VERIFY is the output of AVERIFY algorithm.

*Correctness.*

The correctness of our second scheme is justified as follows.

$$\begin{aligned} & \hat{e}(P_{pub}, U_i + h_i Q_{ID_i} + U_j + h_j Q_{ID_j}) \\ &= \hat{e}(P_{pub}, r'_i Q_{ID_i} - U_j - h_j Q_{ID_j} + h_i Q_{ID_i} + U_j + h_j Q_{ID_j}) \\ &= \hat{e}(sP, (h_i + r'_i) Q_{ID_i}) = \hat{e}(P, (h_i + r'_i) S_{ID_i}) \quad \blacksquare \end{aligned}$$

### 5.1 Security Consideration

**Theorem 5. (Ambiguity)** *Before the keystone  $k$  is released, both signatures are ambiguous.*

**Lemma 2.** *When the output of VERIFY is **accept**, then any third party can be sure who has generated the signature. Any party cannot frame that the other party has signed a message without his or her consent assuming the one-way property of the hash function. This guarantees that the signature is unforgeable.*

**Theorem 6. (Unforgeability)** *The scheme presented in this section is existentially unforgeable under a chosen message attack in the random oracle model, assuming the one-way property of the hash function, the hardness of the discrete logarithm problem and the Co-CDH assumption.*

**Theorem 7. (Fairness)** *For all signatures that are generated with the same keystone will be binding concurrently when the keystone is released.*

**Theorem 8.** *Our ID-based perfect concurrent signature scheme presented in this scheme is secure in the random oracle model, assuming the hardness of the discrete logarithm problem.*

### 5.2 Signature Length and Efficiency

In this scheme, each signature is a three-tuple  $(U_1, U_2, V)$ , where  $U_1, U_2, V \in \mathbb{G}_1$ . With the same setting as our first scheme, our second scheme only requires 1,026 bits or 129 bytes for a pair of signatures. Hence, the signature is nearly as short as that of the first one. This signature length is only one group element on elliptic curve larger than most existing ID-based signature schemes (for example, see the review in [3]). Our second scheme inherits the efficiency of the underlying scheme by Chow *et al.* [8], such that no pairing operation is needed for signing, with a normal computational cost for other algorithms of the protocol.

## 6 Conclusion and Future Research Direction

We introduced the notion of ID-based perfect concurrent signatures, which is an extension of the notion of concurrent signatures proposed in [6]. We provided the first generic construction of (ID-based) perfect concurrent signature protocol in the literature. We presented two concrete constructions of ID-based perfect concurrent signature schemes based on our generic framework. Our second scheme requires no pairing operation in signing. We also provided a complete security analysis for our schemes on their ambiguity, fairness and unforgeability.

Recently, a new ID-based ring signature scheme was proposed [10]. Instead of following the existing paradigms of ID-based ring signature constructions, the scheme is constructed using a cryptographic primitive known as accumulator (e.g. see [10]). It would be interesting to see if concurrent signature could be realized from cryptographic accumulator.

## References

1. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures. *IEEE Journal on Selected Areas in Communications*, 18, 2000.
2. Feng Bao, Robert H. Deng, and Wenbo Mao. Efficient and Practical Fair Exchange Protocols. In *IEEE Symposium on Security and Privacy 1998*, pp. 77–85, 1998.
3. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security Proofs for Identity-Based Identification and Signature Schemes. In *Adv in Cryptology - Eurocrypt 2004, LNCS 3027*, pp. 268–286, 2004.
4. Dan Boneh and Matt Franklin. Identity-based Encryption from the Weil Pairing. In *Adv in Cryptology - Crypto 01, LNCS 2139*, pp. 213–229, 2001.
5. Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Adv in Cryptology - Asiacrypt 2001, LNCS 2248*, pp. 514–532, 2001.
6. Liqun Chen, Caroline Kudla, and Kenneth G. Paterson. Concurrent Signatures. In *Adv in Cryptology - Eurocrypt 2004, LNCS 3027*, pp. 287–305, 2004.
7. Sherman S.M. Chow, Richard W.C. Lui, Lucas C.K. Hui, and Siu Ming Yiu. Identity Based Ring Signature: Why, How and What Next. *EuroPKI 2005, LNCS 3545*, pp. 144–161, 2005.
8. Sherman S.M. Chow, Siu Ming Yiu, and Lucas C.K. Hui. Efficient Identity Based Ring Signature. *Applied Crypto and Network Security - ACNS 2005, LNCS 3531*, pp. 499–512, 2005.
9. Yevgeniy Dodis and Leonid Reyzin. Breaking and Repairing Optimistic Fair Exchange from PODC 2003. *ACM Workshop on Digital Rights Management*, 2003.
10. Lan Nguyen. Accumulators from Bilinear Pairings and Applications. *Topics in Cryptology - CT-RSA 2005, LNCS 3376*, pp. 275–292, 2005.
11. David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. *Adv in Cryptology - Eurocrypt 1996, LNCS 1070*, pp. 387 – 398, 1996.
12. Ronald L. Rivest, Adi Shamir, and Yael Tauman: How to Leak a Secret. *Adv in Cryptology - Asiacrypt 2001, LNCS 2248*, pp. 552 – 565, 2001.
13. Willy Susilo, Yi Mu and Fangguo Zhang. Perfect Concurrent Signature Schemes. *Inf and Comm Security - ICICS 2004, LNCS 3269*, pp. 14–26, 2004.
14. Fangguo Zhang and Kwangjo Kim. ID-based Blind Signature and Ring Signature from Pairings. *Adv in Cryptology - Asiacrypt 2002, LNCS 2501*, pp. 533 – 547, 2002.

# Sequential Aggregate Signatures Working over Independent Homomorphic Trapdoor One-Way Permutation Domains

Huafei Zhu, Feng Bao, and Robert H. Deng

Department of Information Security, I<sup>2</sup>R, A-Star, Singapore 119613  
{huafei, baofeng}@i2r.a-star.edu.sg  
School of Information Systems, Singapore Management University  
robertdeng@smu.edu.sg

**Abstract.** The contribution of this paper has two folds. In the first fold, we propose a generic construction of sequential aggregate signatures from families of certificated trapdoor one-way permutations. We show that our construction is provably secure in the random oracle model assuming that the underlying homomorphic permutations are trapdoor one-way. Compared to Lysyanskaya et al's generic construction that is constructed from a trapdoor one-way permutation family working over the same domain [16], our scheme works over independent trapdoor one-way permutation domains. The flexible choice of the underlying permutation domains benefits our scheme to its applications in the real world where individual user may choose its working domain independently. In the second fold, we instantiate our generic construction with RSA so that the RSA moduli in our scheme can be chosen independently by individual user and thus the moduli is not required to be of the same length. Consequently, our proposed instantiation is the first scheme based on the RSA problem that works for any moduli – this is the most significant feature of our scheme different from the best results constructed from the RSA problem (say, Kawauchi et al's scheme [14], and Lysyanskaya et al's scheme [16]).

**Keywords:** Homomorphic trapdoor one-way permutation, sequential aggregate signature, signature scheme.

## 1 Introduction

In [3], Boneh, Gentry, Lynn and Shacham (BGLS) first introduced and formalized a new notion called aggregate signatures, together with a concrete implementation from the bilinear mapping. An aggregate signature scheme is the following cryptographic primitive: each of  $n$  users with a public/private key pair  $(PK_i, SK_i)$  and a message  $m_i$  wishes to attest to a message  $m_i$ . Informally the procedure can be stated as follows: user  $u_i$  first signs the correspondent message  $m_i$  ( $1 \leq i \leq n$ ) and obtains a signature  $\sigma_i$ , and then  $n$  signatures can be combined by an unrelated party into an aggregate signature. Aggregate signatures are natural extension of multi-signature schemes. In a multi-signature scheme

(e.g., [13], [22], [12], [24], [9], [11], [20] and [21]), a collection of users all sign the same message  $m$ . The result is a single signature (thus the conception of multi-signature schemes is not so good for practical applications since in certain cases we must be able to aggregate signatures on different messages, e.g., Certificate chaining, Secure Border Gateway Protocol). Recently, Micali, Ohta, and Reyzin [18], presented a clear security model and new constructions for multi-signature schemes from Schnorr's signature scheme. Another interesting construction was presented by Boldyreva [1] from the gap Diffie-Hellman assumption.

Burmester et al [2], Doi, Mambo, and Okamoto [8] (DMO), Mitomi and Miyaji [17] have already mentioned that when multiple entities sign a document (hence a set of users all sign the same message), the signing order often reflects the role of each signer and signatures with different signing orders are regarded as different multi-signature schemes. Thus a multi-signature scheme with message flexibility, order flexibility and order verifiability should be required. Burmester et al's then proposed an interesting order-specified multi-signature scheme from modified ElGamal signature scheme while Doi et al's construction is from the RSA problem. Notice that the protocol presented in [8] requires that the public keys corresponding the signing order have to be registered in advance, but it is not necessary in [17]. Later Kawauchi, Komano, Ohta and Tada [14] studied the possibility of simulation of Mitomi and Miyaji's schemes [17] in order to investigate whether that scheme is secure against active attacks. Unfortunately, they showed that Mitomi and Miyaji's scheme cannot be proved secure against active attacks. Tada [25] then proposed an order-specified multi-signature scheme based on the hardness of the discrete logarithm problem. The scheme allows the signing orders to be formed like any series-parallel graphs, which differs from the scheme [20]. The security is shown by using ID-reduction technique, which reduces the security of multi-signature schemes to those of multi-round identification schemes. Finally they constructed alternative RSA-based multi-signature scheme in order to overcome the problem from which Mitomi and Miyaji's scheme suffers.

Very recently, Lysyanskaya et al. [16] presented a clear security model and new constructions for order-specified multi-signature schemes that allow a collection of users to sign different messages (i.e., sequential aggregate signatures). In their paper [16], a generic construction for sequential aggregative signature schemes based on Full Domain Hash, which again is based on trapdoor permutations, is presented. They also instantiated the underlying trapdoor permutations with RSA. Their work is not trivial for several reasons, one being that their generic construction needs the trapdoor permutation to be over the same domain for each user, which is not the case for RSA.

## 1.1 Problem Statement

Sequential aggregate signatures are very useful for many network applications. They can be used in secure border gateway protocol (S-BGP) [15] for securing the UPDATE routing messages and in secure Ad Hoc On-Demand Vector Routing protocol. They can be used in hierarchical public key infrastructure to reduce

the certificate chain as well. To the best of our knowledge, all order-specified multi-signature schemes (or sequential aggregate signatures if each users signs different message) proved to be secure against active attacks are based on either the hardness of the discrete logarithm problem (for example, [18], [20] and [25]) or gap Diffie-Hellman problem (e.g., [1], [3] and [4]) while those based on the RSA problem suffer from unnecessary restrictions (e.g., [14] and [16]). Thus any more satisfactory solution to sequential aggregate signature scheme based on the RSA problem is certainly welcome (the main topic of this paper). Before starting our works, we would like to review the problems among the best results based on RSA below which are closely related to our works:

In [14], Kawauchi, Komano, Ohta and Tada proposed their construction from RSA trapdoor one-way permutations [23]: The key-generation step:  $P_i$  has RSA, which on input  $1^{K_i}$ , randomly selects two distinct  $K_i/2$ -bit primes  $p_i$  and  $q_i$ , and calculates the modulus  $N_i = p_i q_i$ . It randomly picks up an encryption exponent  $e_i \in Z_{\lambda(N_i)}$  and  $\lambda(N_i) = \text{LCM}(p_i - 1, q_i - 1)$ . Also each  $P_i$  uses two hash functions  $G_i$  and  $H_i$ . The first one,  $H_i$ , called the compressor, maps as  $H_i: \{0, 1\}^* \rightarrow \{0, 1\}^{k_2}$  and the other one,  $G_i$ , called the generator, maps as  $G_i: \{0, 1\}^{k_2} \rightarrow \{0, 1\}^{K_{i-1} + k_1}$ , where  $K_i = 1 + K_{i-1} + k_1 + k_2$ ,  $K_0 = k_0$ <sup>1</sup>. Signature generation step (for the same message  $m$ ): Suppose that  $\mathcal{P} = \{P_1, \dots, P_n\}$  generates a multi-signature for a message  $m$ .  $P_1$  picks up an  $r_1 \in_R \{0, 1\}^{k_1}$  to compute  $\omega_1 = H_1(m, r_1)$ . Also  $P_1$  computes  $\sigma_1 = z_1^{d_1} \bmod N_1$ , where  $z_1 = 0 || t_1 || s_1 || \omega_1$ ,  $(t_1 || s_1) = (0^{k_0} || r_1) \oplus G_1(\omega_1)$ . Then  $P_1$  sends  $(m, \sigma_1)$  as a signature for  $m$  to  $P_2$ . For each  $i \in [2, n]$ , the following is executed:  $P_i$  receives  $(m, \sigma_{i-1})$  from  $P_{i-1}$ .  $P_i$  then checks the validity of received signature which is described below. If it is invalid, halts, otherwise,  $P_i$  picks up an  $r_i \in_R \{0, 1\}^{k_1}$  to compute  $\omega_i = H_i(m, r_i, \sigma_{i-1})$ . Also  $P_i$  computes  $\sigma_i = z_i^{d_i} \bmod N_i$ , where  $z_i = 0 || t_i || s_i || \omega_i$ ,  $(t_i || s_i) = (\sigma_{i-1} || r_i) \oplus G_i(\omega_i)$ . Then  $P_i$  sends  $(m, \sigma_i)$  as a signature for  $m$  to  $P_{i+1}$ , where  $P_{n+1}$  is a verifier. Verification step: Suppose that the verifier  $V$  receives  $(m, \sigma_n)$  as a multi-signature for a message  $m$ . For each  $i = n$  to  $i = 2$ , the following is executed by the verifier. -First,  $V$  computes  $z_i = \sigma_i^{e_i} \bmod N_i$ , breaks up  $z_i$  as  $b_i || t_i || s_i || \omega_i$ . (That is, let  $b_i$  be the first bit of  $z_i$ ,  $t_i$  the next  $K_{i-1}$  bits,  $s_i$  the next  $k_1$  bits, and  $\omega_i$  the remaining  $k_2$  bits.) And  $V$  calculates  $(\alpha_i || \beta_i) = (t_i || s_i) \oplus G_i(\omega_i)$ . If  $b_i = 0$  and  $\omega_i = H_i(m, \beta_i, \alpha_i)$ , then  $V$  computes  $\alpha_i = \sigma_{i-1}$  and goes on the step. Finally  $V$  obtains  $\sigma_1$ , computes  $z_1 = \sigma_1^{e_1} \bmod N_1$ , breaks up  $z_1$  as  $b_1 || t_1 || s_1 || \omega_1$ , and calculates  $(\alpha_1 || \beta_1) = (t_1 || s_1) \oplus G_1(\omega_1)$ . If  $b_1 = 0$ ,  $\omega_1 = H_1(m, \beta_1, \alpha_1)$  and  $\alpha_1 = 0^{k_0}$ , then  $V$  returns 1 else return 0.

In [16], Lysyanskaya, Micali, Reyzin, Shacham proposed two approaches to instantiate their generic construction from RSA trapdoor one-way permutations: the first approach is to require the user's moduli to be arranged in increasing order:  $N_1 < N_2 < \dots < N_t$ . At the verification, it is important to check that the  $i$ -th signature  $\sigma_i$  is actually less than  $N_i$  to ensure the signatures are unique if  $H$  is fixed. As long as  $\log(N_1) - \log(N_t)$  is constant, the range of  $H$  is a subset of  $Z_{N_1}$  whose size is the constant fraction of  $N_1$ , the scheme will be secure; the

---

<sup>1</sup> Thus the restriction  $|N_i| - |N_{i-1}| = 1 + k_1 + k_2$  is posted. We see that this unpleasant restriction should be removed from the point of views of practical applications.

second approach does not require the moduli to be arranged in increasing order, however they are required to be of the same length. The signature will be expanded by  $n$  bits  $b_1, \dots, b_n$ , where  $n$  is the total number of users. Namely, during signing, if  $\sigma_i \geq N_{i+1}$ , let  $b_i = 1$ ; else, let  $b_i = 0$ . During the verification, if  $b_i = 1$ , add  $N_{i+1}$  to  $\sigma_i$  before proceeding with the verification of  $\sigma_i$ . Always, check that  $\sigma_i$  is in the correct range  $0 \leq \sigma_i \leq N_i$  to ensure the uniqueness of signatures.

We would like to provide the following remarks on KKOT scheme [14] and Lysyanskaya et al's scheme [16]: Lysyanskaya et al's first scheme can be viewed as an improvement of the KKOT scheme [14]. The restriction of moduli in Tada's scheme  $|N_i| - |N_{i-1}| = 1 + k_1 + k_2$  is weakened by the restriction of users's moduli to be arranged in increasing order  $N_1 < N_2 < \dots < N_t$  in Lysyanskaya et al's scheme. The second approach of Lysyanskaya et al's scheme does not require the moduli to be arranged in increasing order, however they are required to be of the same length and the signature size will be expanded by  $n$  bits  $b_1, \dots, b_n$ , where  $n$  is the total number of users. Namely, during signing, if  $\sigma_i \geq N_{i+1}$ , let  $b_i = 1$ ; else, let  $b_i = 0$ . For applications of sequential aggregate signature schemes in the scenarios discussed above, the choice of  $N_i$  of a host is independent on the choice of another host  $N_j$  in the Internet (in case the underlying protocol is constructed from RSA). A reasonable assumption should be that the sizes of all moduli are bounded by a fixed size. Since there is no solution to this problem, an interesting research problem can be addressed as:

*Research problem: how to construct practical and secure sequential aggregate signatures assuming that all moduli are bounded by a fixed size?*

## 1.2 Our Works

In this paper, we first propose sequential aggregate signatures in which the set of participants is ordered. The aggregate signature is computed by having each signer, in turn, add his signature to it. We propose a generic construction of sequential aggregate signatures from families of certificated trapdoor one-way permutations. We then show that our construction is provably secure in the random oracle model assuming that the underlying homomorphic permutations are trapdoor one-way. Compared to Lysyanskaya et al's generic construction that is constructed from a trapdoor one-way permutation family working over the same domain [16], our scheme works over independent trapdoor one-way permutation domains. The flexible choice of the underlying permutation domains benefits our scheme to its applications in the real world where individual user may choose its working domain independently. Finally, we instantiate our generic construction with RSA that enjoys the following nice features: All three signatures are based on the hardness of the RSA problem. Notice that the computation complexity of our scheme for the  $i$ -th user signing a message needs one exponent computation while the verification processing needs  $(i - 1)$  exponent computations. Thus all three schemes mentioned above have approximate computation complexity; In our scheme, the moduli are not required to be of the same length, i.e., each RSA modulus  $N_i$  is chosen independently by individual user. Thus our construction is the first scheme from RSA that works for any moduli – the most significant feature of our scheme different from all known schemes available in the literature.

## 2 Standard Signature Schemes Working over Extended Domains

### 2.1 Notions

Definition 1: A collection of permutation  $F = \{f_i : D_i \rightarrow D_i \mid i \in I\}$  over some index set  $I \subset \{0, 1\}^k$  is said to be a family of trapdoor one-way permutations if: there is an efficient sampling algorithm  $S(1^k)$  which outputs a random string index  $i \in \{0, 1\}^k \cap I$ , and a trapdoor information  $sk_i$ ; there is an efficient sampling algorithm which, on input  $i$ , outputs a random  $x \in D_i$ . Notice that there must be a mathematical structure associated with  $D_i$ . For simplicity, we assume that  $G_i$  is a group (not necessary a cyclic group, e.g.,  $D_i = Z_{n_i}^*$ , if  $f_i$  is the RSA function); each  $f_i$  is efficiently computable given  $i$  and input  $x \in D_i$ ; each  $f_i$  is efficiently invertible given the trapdoor information  $sk_i$  and output  $y \in D_i$ ; for any probabilistic algorithm  $\mathcal{A}$ ,  $\mathcal{A}$  is said  $(t(k), \epsilon(k))$ -break  $F$ , if  $\mathcal{A}$  runs in time at most  $t(k)$  and  $Adv_{\mathcal{A}}^F(k) \geq \epsilon(k)$ , where the advantage of  $\mathcal{A}$  is defined as  $Adv_{\mathcal{A}}^F(k) = \Pr[x' = x \mid (i, sk_i) \leftarrow S(1^k), x \leftarrow G_i, y = f_i(x), x' \leftarrow \mathcal{A}(i, y)]$ .  $F$  is said to be  $(t(k), \epsilon(k))$ -secure if no adversary  $\mathcal{A}$  can  $(t(k), \epsilon(k))$ -break it.

Definition 2: A collection of trapdoor one-way permutation  $F = \{f_i : D_i \rightarrow D_i \mid i \in I\}$  over some index set  $I \subset \{0, 1\}^k$  is said to be homomorphic if for any  $(D_i, f_i) \leftarrow i$  and  $x, y \in D_i$ , it satisfies  $f_i(xy) = f_i(x)f_i(y)$ .

We review the well known definition of security of ordinary digital signatures [10]. Existential unforgeability under a chosen message attack in the random oracle [5] for a signature scheme  $(KG, Sig, Vf)$  with a random oracle is defined using the following game between a challenger and an *adv* (notice that this security definition can also be applied to the scenario where a collection of random oracles are deployed): the challenger runs  $KG$  to obtain a public key  $pk$  and private key  $sk$ . The adversary *adv* is given  $pk$ ; Proceeding adaptively, *adv* requests signatures with  $pk$  on at most  $q_{sig}$  messages of his choice  $m_1, \dots, m_{q_{sig}}$ . The challenge responds to each query with a signature  $\sigma_i$ . Algorithm *adv* also adaptively asks for at most  $q_H$  queries of the random oracle  $H$ ; *adv* outputs a pair  $(m, \sigma)$  and wins the game if  $m \notin \{m_1, \dots, m_{q_{sig}}\}$ , and  $Vf(pk, m, \sigma) = 1$  (a valid signature of the message  $m$ ).

By  $AdvSig_{\mathcal{A}}$ , we denote the probability of success of an adversary.

Definition 3: We say a signature scheme is secure against adaptive chosen-message attack if for every polynomial time Turing machine  $\mathcal{A}$ , the probability  $AdvSig_{\mathcal{A}}$  that it wins the game is at most a negligible amount, where the probability is taken over coin tosses of  $KG$  and  $Sig$  and  $\mathcal{A}$ .

### 2.2 Generic Construction

We show that our signature scheme constructed from the extended domain of homomorphic trapdoor one-way permutations is provably secure against adaptive chosen message attack in the random oracle model [5]. We assume that a



permutation used to construct our sequential aggregate signature scheme must be a certificated one-trapdoor permutation. A certified trapdoor one-way permutation is one such that, for any describing string  $\text{des}$ , it is easy to determine whether  $\text{des}$  can have been output by a trapdoor one-way permutation generator, and thereby ensure that  $f(\text{des}, \cdot)$  is a permutation.

-Key generation algorithm  $KG$ : On input a security parameter  $l, k$ ,  $KG$  specifies two cryptographic hash functions  $H: \{0, 1\}^* \rightarrow \{0, 1\}^l$  and  $G: \{0, 1\}^t \rightarrow \{0, 1\}^k$ ,  $t = l - k$ . On input  $k$ ,  $KG$  outputs an instance of homomorphic trapdoor one-way permutation  $\{f : D \rightarrow D\}$ . We assume that each element in  $D$  can be represented by a  $k$ -bit string, i.e.,  $D \subset \{0, 1\}^k$ . We further assume that there is an efficiently computable mapping from  $\{0, 1\}^k \setminus D$  to  $D$  and given  $\tau(x)$  and  $b$ , it is easy for one to recover  $x \in \{0, 1\}^k$ , where  $\tau(x)$  maps a element  $x \in \{0, 1\}^k$  to  $x$  modulo  $|D|$ , i.e.,  $\tau^0(x) = x$  if  $x \in D$  while  $\tau^1(x) = x \bmod |D|$ , if  $x \in \{0, 1\}^k \setminus D$  (it will be clear in case that the underlying homomorphic one-way trapdoor permutation is instantiated with RSA, see next for more details).

-Signing algorithm: On input a message  $m$ , it computes  $x = H(m)$  and then separates  $x = y||z$ , where  $y \in \{0, 1\}^k$  and  $z \in \{0, 1\}^t$ ,  $t = l - k$ . Finally, it computes  $g = f^{-1}(\tau^b(y \oplus G(z)))||z$ , where  $\tau$  is an efficient mapping from  $\{0, 1\}^k \setminus D$  to  $D$ . That is, if  $y \oplus G(z) \in D$ , then the signature  $\sigma$  of the message  $m$  is  $(g, 0)$  ( $b=0$ ); if  $y \oplus G(z) \in \{0, 1\}^k \setminus D$ , then the signature  $\sigma$  of the message  $m$  is denoted by  $(g, 1)$  ( $b=1$ ).

-The verification algorithm is given as input a signature  $\sigma = (g, b)$ , the messages  $m$ , and the correspondent public key  $pk$  and proceeds as follows: first it computes  $x = H(m)$  and separates  $x = y||z$  and  $g = v||w$  and checks whether  $w$  is  $z$ , if not, it outputs 0; Else, it checks that  $pk$  and  $f$  is a certificated permutation. If both conditions are valid, then it computes  $y$  from the equation  $\tau^b(y \oplus G(z)) = f(v)$ . And output 1, if the test  $H(m) = y||z$  is valid.

Lemma 1: Let  $f$  be a homomorphic trapdoor one-way permutation defined over  $D$ , and  $\tau$  be an efficiently computable mapping from  $\{0, 1\}^k \setminus D$  to  $D$  and given  $\tau(x)$ , it is easy for one to recover  $x \in \{0, 1\}^k$ , then our signature scheme is secure within the random oracle model.

Proof: We follow Coron's full domain reduction [7]. Let  $F$  be a forger that  $(t, q_{sig}, q_H, q_G, \epsilon)$  breaks our sequential aggregate signature scheme. We assume that  $F$  never repeats a hash query and a signature query. We will build an inverter that can  $(t', \epsilon')$  breaks underlying one-way trapdoor permutation. The inverter receives an input  $D, f$ , where  $D$  and  $f$  are public keys and  $Y \in D$  is chosen uniformly at random. The inverter tries to find  $X = f^{-1}(Y)$ . The inverter starts running  $F$  and makes hash oracle queries and signing queries on behalf of the inverter. The inverter will answer hash oracle queries ( $H$ -oracle query and  $G$ -oracle query) and signing oracle queries. We assume for simplicity that when  $F$  requests a signature of the message  $m$ , it has already made the corresponding hash queries on  $m$ . If not, the inverter goes ahead to make  $H$ -oracle query and then to make  $G$ -oracle query. The inverter uses counter  $i$  initially set to zero. When  $F$  makes a  $H$ -oracle for  $m$ , the inverter increments  $i$ , sets  $m_i = m$  and picks two random strings  $y_i \in \{0, 1\}^k$  and  $z_i \in \{0, 1\}^{l-k}$ , where  $k$  is the bit length

of  $D$ . The output of  $H$ -oracle is  $x_i$  ( $x_i = y_i || z_i$ ). To make the  $G$ -oracle query of the string  $z_i$ , the inverter first checks that whether  $z_i$  has been made the  $G$ -oracle query, and then the inverter will look at the  $H$ -oracle table for any query  $m_i$  queried to  $H$ -oracle with answer  $(y_i, z_i)$ . It will output the string  $v_i$  which is defined below if  $z_i$  has already requested (notice that since  $H$  is a random oracle, the probability that the  $H$ -oracle will output  $(*, z_i)$  for any message different than  $m_i$  is negligible assuming that the amount  $1/2^{(l-t)}$  is negligible). The inverter now picks a random string  $r_i \in D$  then returns  $v_i$  such that  $\tau^b(y_i \oplus v_i) = f(r_i)$  with probability  $p$  and  $v_i$  such that  $\tau^b(v_i \oplus y_i) = Yf(r_i)$  with probability  $(1 - p)$ . Here  $p$  is a fixed probability which will be determined later. When  $F$  makes a signing query for  $m$ , it has already requested the hash queries of  $m$ , so  $m = m_i$  for some  $i$ . If  $\tau^b(v_i \oplus y_i) = f(r_i)$ , then the inverter returns  $r_i$  as the signature, otherwise the process stop and the inverter has failed. Eventually,  $F$  halts and outputs a forgery  $(m, \sigma)$ . We assume that  $m$  has requested  $H$ -oracle and  $G$ -oracle of  $m$  before. If not,  $I$  goes ahead and makes the hash oracle queries itself, so that in any case,  $m = m_i$  for some  $i$ . Then if  $\tau^b(v_i \oplus y_i) = Yf(r_i)$ , we can compute  $f^{-1}(Y) = \tau^b(v_i \oplus y_i) r_i$  with probability  $p^{q_{sig}}(1 - p)$ . Setting  $p = 1 - \frac{1}{q_{sig} + 1}$ , it follows that the probability that the inverter can find  $y \in D$  such that  $f(y) = Y$  with probability  $\epsilon = exp(1)q_{sig}\epsilon'$  for sufficiently large  $q_{sig}$ .

### 2.3 Instantiated with RSA

Specifically to RSA instance [23], a certificated permutation could be done by having a trusted certification authority to check that  $N$  is a product of two large primes and  $e$  is relative prime to  $\phi(N)$  before issuing a certificate. This check, however, requires one to place more trust in the authority than usual. Namely, the authority must be trusted not just to verify the identity of a key's purported owner, but also to perform verification of some complicated properties of the key. More precisely,

Lemma 2: suppose  $\gcd(e, p - 1) = k_1 k_2$ ,  $\gcd(e, q - 1) = k_2 k_3$ ,  $\gcd(k_1, q - 1) = 1$ ,  $\gcd(k_3, p - 1) = 1$  (i.e., we consider the case where  $k_1$  is a factor of  $p - 1$ ,  $k_2$  is a common divisor of  $p - 1$  and  $q - 1$ ,  $k_3$  is a factor of  $q - 1$  and  $k_1, k_2$  and  $k_3$  are pair wise prime), then the number of set  $A$  is  $k_1 k_2 k_3$ .

Proof: We consider the following three cases.

- case 1:  $\gcd(e, p - 1) = k \neq 1$ ,  $\gcd(e, q - 1) = 1$ ; Since  $f(x) = x^e \pmod q$  is a permutation from  $Z_q^*$  to  $Z_q^*$ , we will consider  $g(x) = x^e \pmod p$  from  $Z_p^*$  to  $Z_p^*$ . Let  $g$  be a generator of  $Z_p^*$  and denote  $B = \{g^k : x \in Z_p^*\}$ . Since  $\gcd(e, p - 1) = k \neq 1$ , it follows that  $\gcd(e/k, p - 1) = 1$ . Denote  $g(x) = x^e \pmod p = g_1(g_2(x))$ , where  $g_1(x) = x^k \pmod p$  and  $g_2(x) = x^{e/k} \pmod p$ . Notice that  $g_2$  is a permutation from  $Z_p^*$  to  $Z_p^*$  but  $g_1(x)$  is a homomorphism from  $Z_p^*$  to  $Z_p^*$ . Since order  $\text{ord}(g^k) = \frac{p-1}{\gcd(k, p-1)}$ , it follows that the number of elements of  $B$  is  $k$  and so the number of the set  $A$  is also  $k$ .
- case 2:  $\gcd(e, p - 1) = 1$ ,  $\gcd(e, q - 1) = k \neq 1$ ; Same claim as case 1.

- case 3:  $k = k_1 k_2 k_3$ ,  $\gcd(e, p-1) = k_1 k_2$ ,  $\gcd(e, q-1) = k_2 k_3$ ,  $\gcd(k_1, q-1) = 1$ ,  $\gcd(k_3, p-1) = 1$  (i.e., we consider the case where  $k_1$  is a factor of  $p-1$ ,  $k_2$  is a common divisor of  $p-1$  and  $q-1$ ,  $k_3$  is a factor of  $q-1$  and  $k_1, k_2$  and  $k_3$  are pair wise prime). Since  $Z_n^*$  and  $Z_p^* \times Z_q^*$  are isomorphic and  $f(x) = x^{e/k} \pmod n$  is a permutation from  $Z_p^* \times Z_q^*$  to  $Z_p^* \times Z_q^*$ , we will only consider the function  $g(x) = x^k \pmod n$ . Denote  $f_i(x) = x^{k_i} \pmod n$ , then  $g(x) = f_3(f_2(f_1(x))) = x^k \pmod n$ . Denote  $B = \{x^k : x \in Z_p^* \times Z_q^*\}$ . We know that there are  $k_1 k_2$  elements  $x_1 \in Z_p^*$  such that  $x_1^{k_1} = 1$  (this 1 is in  $Z_p^*$ ). And there are  $k_2 k_3$  elements  $x_2 \in Z_q^*$  such that  $x_2^{k_2} = 1$  (this 1 is in  $Z_q^*$ ). So the number of the set  $A$  is  $k_1 k_2 k_3$ .

Alternative approach may allow one to choose a large prime number  $e$  such that  $e > N$ , and then to show that  $e$  is a prime number by making use of the prime test protocol. This approach is attractive and has been used in [16]. Our sequential aggregate signature scheme will make use of this approach.

We show that our signature scheme described below is provably secure against adaptive chosen message attack in the random oracle model [5] assuming that the RSA problem (on input a randomly chosen  $y \in Z_N^*$ , and the public key  $(e, N)$ , outputs  $x \in Z_N^*$  such that  $y = x^e \pmod N$ ) is hard.

-Key generation algorithm: On input a security parameter  $k$ , it generates an RSA public key  $(N, e)$  and secret key  $(N, d)$ , ensuring that  $|N| = k$ -bit and that  $e > N$  is a prime. Let  $f^{-1}(x) = x^d \pmod N$  be the inverse function of RSA function  $f(x) = x^e \pmod N$  ( $ed \equiv 1 \pmod \phi(N)$ ). On input  $l$  and  $k$ , it also specifies two cryptographic hash functions  $H: \{0, 1\}^* \rightarrow \{0, 1\}^l$  and  $G: \{0, 1\}^t \rightarrow \{0, 1\}^k$ ,  $t = l - k$ .

-Signing algorithm: On input a message  $m$ , it computes  $x = H(m)$  and then separates  $x = y||z$ , where  $y \in \{0, 1\}^k$  and  $z \in \{0, 1\}^t$ ,  $t = l - k$ . Finally, it computes  $g = f^{-1}(y \oplus G(z))||z$ . If  $y \oplus G(z) > N$ , then the signature  $\sigma$  of the message  $m$  is  $(g, b)$ , where  $b = 1$ ; if  $y \oplus G(z) < N$ , then the signature  $\sigma$  of the message  $m$  is denoted by  $(g, b)$ , where  $b = 0$  (in this case  $\tau(x) = x - bN$  in the RSA instantiation); Note that the probability that the event  $y \oplus G(z) = N$  happens is at most negligible amount, therefore we ignore this event in the following discussions.

-The verification is given as input a signature  $\sigma = (g, b)$ , the messages  $m$ , and the correspondent public key  $(N, e)$  and proceeds as follows: first it computes  $x = H(m)$  and separates  $x = y||z$  and  $g = v||w$  and checks whether  $w$  is  $z$ , if not, it outputs 0; Else, it checks that  $e > N$  and  $e$  is a prime number. If both conditions are valid, then it checks the validity of the equation  $y = \mathcal{B}(f(v) + bN) \oplus G(z)$ , where  $\mathcal{B}(x)$  is the binary representation of  $x \in \mathcal{Z}$ . And output 1, if the equation is valid.

At first glance it seems that the adversary may have choice whether to use  $b = 0$  or  $b = 1$ . However, this will result in two values  $y \oplus G(z)$  that are guaranteed to be different: one is less than  $N$  and the other at least  $N$ . Hence uniqueness of  $\sigma$  implies uniqueness of  $b$ . Notice that once  $m$  is given, the value  $y \oplus G(z)$  is determined assuming that  $H(m)$  and  $G(z)$  have already been queried. Furthermore, since the functionality of bit  $b$  defined above is to identify whether

$y \oplus G(z) > N$  or not, and  $f(y \oplus G(z) + N) = f(y \oplus G(z))$  (an invariant of RSA function  $f(x) = x^e \bmod n$ ), we can simply assume that  $y \oplus G(z) < N$  in the following security argument. As an immediate application of Lemma 1, we have the following statement:

Corollary 1: Under the hardness of the RSA problem, the ordinary signature scheme described above is secure against adaptive chosen message attack in the random oracle model in the sense of [10].

### 3 Syntax, Security Definition, Construction of Sequential Aggregate Signature Scheme from Ordinary Signatures

#### 3.1 Syntax

A sequential signature scheme (KG, AggSign, AggVf) consists of the following algorithms: Key generation algorithm (KG): On input  $l$  and  $k_i$ , KG outputs system parameters **param** (including an initial value  $\mathcal{IV}$ , without loss of generality, we assume that  $\mathcal{IV}$  is a zero strings with length  $l$ -bit), on input **param** and user index  $i \in \mathcal{I}$  and  $k_i$ , it outputs a public key and secret key pair  $(PK_i, SK_i)$  of a trapdoor one-way permutation  $f_i$  for a user  $i$ . Aggregate signing algorithm (AggSign): Given a message  $m_i$  to sign, and a sequential aggregate  $\sigma_{i-1}$  on messages  $\{m_1, \dots, m_{i-1}\}$  under respective public keys  $PK_1, \dots, PK_{i-1}$ , where  $m_1$  is the inmost message. All of  $m_1, \dots, m_{i-1}$  and  $PK_1, \dots, PK_{i-1}$  must be provided as inputs. AggSign first verifies that  $\sigma_{i-1}$  is a valid aggregate for messages  $\{m_1, \dots, m_{i-1}\}$  using the verification algorithm defined below (if  $i=1$ , the aggregate  $\sigma_0$  is taken to be zero strings  $0^l$ ). If not, it outputs  $\perp$ , otherwise, it then adds a signature on  $m_i$  under  $SK_i$  to the aggregate and outputs a sequential aggregate  $\sigma_i$  on all  $i$  messages  $m_1, \dots, m_i$ . Aggregate verifying algorithm (AggVf): Given a sequential aggregate signature  $\sigma_i$  on the messages  $\{m_1, \dots, m_i\}$  under the respective public keys  $\{PK_1, \dots, PK_i\}$ . If any key appears twice, if any element  $PK_i$  does not describe a permutation or if the size of the messages is different from the size of the respective public keys reject. Otherwise, for  $j = i, \dots, 1$ , set  $\sigma_{j-1} = f_j(PK_1, \dots, PK_j, \sigma_j)$ . The verification of  $\sigma_{i-1}$  is processed recursively. The base case for recursion is  $i = 0$ , in which case simply check that  $\sigma_0$ . Accepts if  $\sigma_0$  equals the zero strings.

#### 3.2 The Definition of Security

The following security definition of sequential aggregative signature schemes is due to [16]. The aggregate forger  $\mathcal{A}$  is provided with a initial value  $\mathcal{IV}$ , a set of public keys  $PK_1, \dots, PK_{i-1}$  and  $PK$ , generated at random. The adversary also is provided with  $SK_1, \dots, SK_{i-1}$ ;  $PK$  is called target public key.  $\mathcal{A}$  requests sequential aggregate signatures with  $PK$  on messages of his choice. For each query, he supplies a sequential aggregate signature  $\sigma_{i-1}$  on some messages  $m_1, \dots, m_{i-1}$  under the distinct public keys  $PK_1, \dots, PK_{i-1}$ , and an additional message  $m_i$  to be signed by the signing oracle under public key  $PK$ . Finally,

$\mathcal{A}$  outputs a valid signature  $\sigma_i$  of a message  $m_i$  which is associated with the aggregate  $\sigma_{i-1}$ . The forger wins if  $\mathcal{A}$  did not request  $(m_i, \sigma_{i-1})$  in the previous signing oracle queries. By  $\text{AdvAggSign}_{\mathcal{A}}$ , we denote the probability of success of an adversary.

Definition 4: We say a sequential aggregate signature scheme is secure against adaptive chosen-message attack if for every polynomial time Turing machine  $\mathcal{A}$ , the probability  $\text{AdvAggSign}_{\mathcal{A}}$  that it wins the game is at most a negligible amount, where the probability is taken over coin tosses of KG and AggSign and  $\mathcal{A}$ .

### 3.3 Generic Construction from Independent Homomorphic Trapdoor One-Way Permutations

We now propose an interesting method to construct aggregate signature schemes from independent homomorphic trapdoor one-way permutations.

-Key generation: each participant  $i$  runs its key generation algorithm  $KG_i$  on input  $l, k_i$ ,  $KG_i$  specifies two cryptographic hash functions  $H: \{0, 1\}^* \rightarrow \{0, 1\}^l$  and  $G_i: \{0, 1\}^{t_i} \rightarrow \{0, 1\}^{k_i}$ ,  $t_i=l - k_i$  (notice that the system security parameter  $l$  should be shared by all participants). On input  $k_i$ ,  $KG_i$  outputs an instance of homomorphic trapdoor one-way permutation  $\{f_i : D_i \rightarrow D_i\}$ . We assume that each element in  $D_i$  can be represented by a  $k_i$ -bit string, i.e.,  $D_i \subset \{0, 1\}^{k_i}$ . We further assume that there is an efficiently computable mapping from  $\{0, 1\}^{k_i} \setminus D_i$  to  $D_i$  and given  $\tau(x)$  and  $b$ , it is easy for one to recover  $x \in \{0, 1\}^{k_i}$ . The public key  $pk_i$  is  $(f_i, D_i, G_i, H)$ ; The private key  $sk_i$  is the trapdoor information of  $f_i$ ;

-Aggregate signing: the input is a private key  $sk_i$ , a message  $m \in \{0, 1\}^*$  to be signed, and a sequential aggregate  $\sigma_{i-1} = (g_{i-1}, b_1, \dots, b_{i-1})$  on messages  $\mathbf{m}|_1^{i-1}$ , under the public keys  $\mathbf{pk}|_1^{i-1}$  (for a vector  $\mathbf{x}$ , we denote a sub-vector containing  $x_a, \dots, x_b$  by  $\mathbf{x}|_a^b$ ). Verify that  $\sigma'$  is a valid signature on  $\mathbf{m}$  under  $\mathbf{pk}$  using the verification algorithm below; if not, output  $\perp$  indicating error. Otherwise, compute  $h_i \leftarrow H(\mathbf{pk}|_1^i, \mathbf{m}|_1^i)$ . The signer then rewrites  $h_i \oplus g_{i-1} = x_i := y_i || z_i$ , and computes  $g_i = f_i^{-1}(\tau^{b_i}(y_i \oplus G_i(z_i))) || z_i$ . The signature of  $\mathbf{m}|_1^i$  under  $\mathbf{pk}|_1^i$  is denoted by  $\sigma_i = (g_i, b_1, \dots, b_i)$ ;

-Aggregate verification: the input is a sequential aggregate  $\sigma_i$  on message  $\mathbf{m}|_1^i$  under  $\mathbf{pk}|_1^i$ . If any public key appears twice in  $\mathbf{pk}|_1^i$ , if any element of  $\mathbf{pk}|_1^i$  does not describe a valid permutation, reject; Otherwise, for  $j=i, \dots, 1$ , the verification algorithm processes the following steps recursively:

- for a given  $\sigma_i = (g_i, b_i)$ , setting  $v_i || w_i \leftarrow g_i$ ,  $z_i \leftarrow w_i$ ;
- computing  $y_i$  from the equation  $\tau^{b_i}(y_i \oplus w_i) = f(v_i)$ ; and setting  $x_i \leftarrow y_i || z_i$ ;
- computing  $h_i \leftarrow H(\mathbf{pk}|_1^i, \mathbf{m}|_1^i)$ , and then  $g_{i-1} \leftarrow x_i \oplus h_i$ .

-Accept if  $\sigma_0$  is equal to  $0^l$  (the initial value of sequential aggregate signature scheme);

### 3.4 The Proof of Security

Theorem: Let  $\cup_{i \in I} f_i$  be a certificated homomorphic trapdoor permutation family. Then our sequential aggregate signature scheme described above is secure in the random oracle model.

Proof: Suppose  $adv$  is a forger algorithm that with non-negligible probability  $\epsilon$  breaks the sequential aggregate signature scheme. We construct an algorithm  $F$  that inverts the permutation given by  $pk_i$  on a given input  $z \in D_i$  which is chosen uniformly at random. Recall that the security definition of a sequential aggregate signature scheme allows an adversary to generate a collection of public keys  $pk_j$  ( $j = 1, \dots, i$ ), and obtain the correspondent  $sk_j$  except for the trapdoor information  $sk_i$  of a target permutation. Thus, the security of the sequential aggregate signature scheme can be reduced to that of the underlying (ordinary) signature scheme. Since the simulator knows  $sk_j$  for  $j \neq i$ , it follows that the simulation of the  $j$ -th user can be trivially simulated while the simulation of  $i$ -th user can be simulated exactly as that described in the proof Lemma 1. Consequently, the proof of security of the theorem follows from an immediate application of Lemma 1.

### 3.5 Instantiated with RSA

Let  $H: \{0, 1\}^* \rightarrow \{0, 1\}^l$  be a cryptographic hash function and  $\mathcal{TV}$  be the initial vector that should be pre-described by a sequential aggregate signature scheme. The initial value could be a random  $l$ -bit string or an empty string. Without loss of generality, we assume that the initial value  $\mathcal{TV}$  is  $0^l$ . Our sequential aggregate signature scheme is described as follows:

-Key generation: Each user  $i$  generates an RSA public key  $(N_i, e_i)$  and secret key  $(N_i, d_i)$ , ensuring that  $|N_i| = k_i$  and that  $e_i > N_i$  is a prime. Let  $G_i: \{0, 1\}^{t_i} \rightarrow \{0, 1\}^{k_i}$ , be cryptographic hash function specified by each user  $i$ ,  $t_i = l - k_i$ .

-AggSig: User  $i$  is given an aggregate signature  $g_{i-1}$  and  $(b_1, \dots, b_{i-1})$ , a sequence of messages  $m_1, \dots, m_{i-1}$ , and the corresponding keys  $(N_1, e_1), \dots, (N_{i-1}, e_{i-1})$ . User  $i$  first verifies  $\sigma_{i-1}$ , using the verification procedure below, where  $\sigma_0 = 0^l$ . If this succeeds, user  $i$  computes  $H_i = H(m_1, \dots, m_i, (N_1, e_1), \dots, (N_i, e_i))$  and computes  $x_i = H_i \oplus g_{i-1}$ . Then it separates  $x_i = y_i || z_i$ , where  $y_i \in \{0, 1\}^{k_i}$  and  $z_i \in \{0, 1\}^{t_i}$ ,  $t_i = l - k_i$ . Finally, it computes  $g_i = f_i^{-1}(y_i \oplus G_i(z_i)) || z_i$ . By  $\sigma_i \leftarrow (g_i, b_i)$ , we denote the aggregate signature (if  $y_i \oplus G_i(z_i) > N_i$ , then  $b_i = 1$ , if  $y_i \oplus G_i(z_i) < N_i$ , then  $b_i = 0$ ; again we do not define the case  $y_i \oplus G_i(z_i) = N_i$  since the probability the event happens is negligible), where  $f_i^{-1}(y) = y^{d_i} \bmod N_i$ , the inverse of the RSA function  $f_i(y) = y^{e_i} \bmod N_i$  defined over the domain  $Z_{N_i}^*$ .

-AggVf: The verification is given as input an aggregate signature  $g_i, (b_1, \dots, b_i)$ , the messages  $m_1, \dots, m_i$ , the correspondent public keys  $(N_1, e_1), \dots, (N_i, e_i)$  and proceeds as follows. Check that no keys appears twice, that  $e_i > N_i$  is a prime. Then it computes:

- $H_i = H(m_1, \dots, m_i, (N_1, e_1), \dots, (N_i, e_i))$ ;
- Separating  $g_i = v_i || w_i$ ;

- Recovering  $x_i$  from the trapdoor one-way permutation by computing  $z_i \leftarrow w_i$ ,  $y_i = \mathcal{B}_i(f_i(v_i) + b_i N_i) \oplus G_i(z_i)$ , and  $x_i = y_i || z_i$ , where  $\mathcal{B}_i(x)$  is the binary representation of  $x \in \mathcal{Z}$  (with  $k_i$  bits).
- Recovering  $g_{i-1}$  by computing  $x_i \oplus H_i$ . The verification of  $(g_{i-1}, b_{i-1})$  is processed recursively. The base case for recursion is  $i = 0$ , in which case simply check that  $\sigma_0 = 0^l$ .

Corollary 2: Our sequential aggregate signature scheme described above is secure in the sense of [16] in the random oracle model.

## 4 Conclusion

In this paper, a generic construction of sequential aggregate signatures has been constructed from homomorphic trapdoor one-way permutations. We have shown that our generic constructions are provably secure in the random oracle model assuming that the underlying homomorphic permutations are trapdoor one-way. We then instantiate our generic constructions with RSA. Compared the best results in the literature, say Kawauchi et al's scheme, and Lysyanskaya et al's scheme [16], our protocol has nice feature: the moduli are not required to be of the same length in our scheme, i.e., in our scheme  $N_i$  is chosen by each user independently. Thus we have proposed the first sequential aggregate signature scheme from RSA that works for any moduli.

## References

1. A. Boldyreva. Efficient threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, Proceedings of PKC 2003, volume 2567 of LNCS, pages 31C46. Springer-Verlag, 2003.
2. M. Burmester, Y. Desmedt, H. Doi, M. Mambo, E. Okamoto, M. Tada, Y. Yoshifuji: A Structured ElGamal-Type Multisignature Scheme. Public Key Cryptography 2000: 466-483
3. Dan Boneh, Craig Gentry, Ben Lynn, Hovav Shacham: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. EUROCRYPT 2003: 416-432.
4. Dan Boneh, Craig Gentry, Ben Lynn, Hovav Shacham: A Survey of Two Signature Aggregation Techniques. In CryptoBytes Vol. 6, No. 2, 2003.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. Denning, R. Pyle, R. Ganesan, R. Sandhu, and V. Ashby, editors, Proceedings of CCS 1993, pages 62-73. ACM Press, 1993.
6. Jan Camenisch, Markus Michels: Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes. EUROCRYPT 1999: 107-122.
7. J. Coron: On the Exact Security of Full Domain Hash. CRYPTO 2000: 229-235.
8. H. Doi, M. Mambo, E. Okamoto: On the Security of the RSA-Based Multisignature Scheme for Various Group Structures. ACISP 2000: 352-367.
9. H. Doi, E. Okamoto, M. Mambo, and T. Uyematsu, Multisignature Scheme with Specified Order, Proc. of the 1994 Symposium on Cryptography and Information Security, SCIS94-2A, January 27 -29, 1994.

10. Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2): 281-308 (1988).
11. P. Horster, M. Michels, and H. Petersen Meta-multisignature schemes based on the discrete logarithm problem, *Information Security -the Next Decade, Proc. of IFIP Sec95*, Chapman-Hall pp. 128 -142 1995.
12. T. Hardjono, and Y. Zheng A practical digital multisignature scheme based on discrete logarithms, *Lecture Notes in Computer Science 718, Proc. of Auscrypt92*, Springer-Verlag, pp. 122-132, 1993.
13. K. Itakura and K. Nakamura. A public key cryptographic suitable for digital multisignatures. *NEC Rearch and Development (71)*, page 1-8, 1983.
14. K. Kawauchi, Y. Komano, K. Ohta and M. Tada: Probabilistic multi-signature schemes using a one-way trapdoor permutation, *IEICE transactions on fundamentals*, vol.E87-A, no5, pp.1141 -1153, 2004. Previous version: Kei Kawauchi, Mitsuru Tada: On the Extract Security of Multi-signature Schemes Based on RSA. *ACISP 2003*: 336-349
15. S. Kent, C. Lynn and K. Seo: Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communicaitons*, Vol. 18, No. 4, Apr. 2000.
16. Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, Hovav Shacham: Sequential Aggregate Signatures from trapdoor one-way permutations. *EUROCRYPT 2004*: 74-90.
17. S. Mitomi and A. Miyaji, "A general model of multisignature schemes with message flexibility, order flexibility, and order verifiability", *IEICE Trans., Fundamentals*. vol. E84-A, No.10(2001), 2488 - 2499. Previous version: S. Mitomi and A. Miyaji, A multisignature scheme with message flexibility, order flexibility and order verifiability, *Information security and privacy-Proceedings of ACISP 2000, Lecture Notes in Computer Science*, 1841(2000), Springer-Verlag, p298 - 312.
18. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures (extended abstract). In *Proceedings of CCS 2001*, pages 245 -254. ACM Press, 2001.
19. K. Ohta, and T. Okamoto, A digital multisignature scheme based on the Fiat-Shamir scheme, *Lecture Notes in Computer Science 739, Advances in Cryptology -Asiacrypt'91*, Springer-Verlag, pp. 139-148, 1993.
20. K. Ohta and T. Okamoto. Multisignature schemes secure against active insider attacks. *IEICE Trans. Fundamentals*, E82-A(1):21C31, 1999.
21. K. Ohta and T. Okamoto: Generic construction methods of multi-signature schemes, *Proceedings of The 2001 Symposium on Cryptography and Information Security (SCIS2001)*, vol.I, pp.31-36, 2001.
22. T. Okamoto. A digital multisignature scheme using bijective public-key cryptosystems. *ACM Trans. Computer Systems*, 6(4):432C41, November 1988.
23. R. Rivest, A. Shamir, and L.M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* 21(2): 120-126 (1978).
24. A. Shimbo, Multisignature Schemes Based on the ElGamal Scheme, *Proc. of The 1994 Symposium on Cryptography and Information Security*, January 27 - 29, 1994.
25. M. Tada: A secure multisignature scheme with signing order Verifiability, *IEICE transactions on fundamentals*, vol.E86-A, no.1, pp.73-88, 2003. Previous version: M. Tada: An Order-Specified Multisignature Scheme Secure against Active Insider Attacks. *ACISP 2002*: 328-345.



# Session Table Architecture for Defending SYN Flood Attack

Xin Li, Zhenzhou Ji, and Mingzeng Hu

School of Computer Science and Technology,  
Harbin Institute of Technology, Harbin, China  
{lixin, jzz, mzhu}@pact518.hit.edu.cn

**Abstract.** Stateful Inspection has become a classical technology for network firewall. Existing session table architectures of Stateful Inspection firewalls cause high time cost of timeout processing. A new architecture is proposed. The new architecture divides a session entry into two separate parts, and designs different data structures for each other. On the base of multi-queue architecture, dynamical timeouts according to available resource improve securities of protected hosts against SYN flood attack. Experimental results show that the new architecture can work well in Gigabit Ethernet network.

## 1 Introduction

Stateful Inspection refers to an extension of packet-by-packet filtering process that tracks individual flows, enabling policy checks that extend across series of packets [1]. Many firewalls have implemented Stateful Inspection technology, such as Cisco PIX [2], 3COM Secure Gateway [3], Netsreen Firewall [1] and Checkpoint FW-1 [4]. Stateful Inspection requires a session table whose entries typically record source and destination IP addresses and port numbers. For each packet, the session table is looked up for a match. A session entry in the format <src-addr, src-port, dst-addr, dst-port, ip-p, state, time> is created when the first packet appears from a flow previously not tracked. Subsequent packets of an established session are checked against the session table rather than against the Rule Base. The performance of Stateful Inspection firewall mainly depends on the performance of processing session table.

Under normal operating condition, all session table entries represent a valid flow. However, abnormal events can create an excessive number of invalid entries in the table. A representative example is the DoS attack using TCP SYN packets. Suppose TCP SYN packets from typically spoofed source addresses go out through a Stateful Inspection firewall in avalanche, triggering the creation of corresponding session entries in the table. When the session table is inundated with invalid entries, performance and security become concerns. So timeout processing is essential for TCP to purge the inactive session [5]. Firewalls are better suited to fight the attack because they tend to be designed to examine packets and maintain connection and state information of session traffic [6].

The TCP connection setup delay is defined to be the time elapsed between the transmission of the first SYN packet and the reception of the corresponding ACK.

The connection setup delay is usually much less than 1 second. At 1 second, more than 93% of the connections are established [5]. Furthermore, after a connection is established, the timeout is automatically set to 3600 seconds, which may cause session table quickly filled up. So the default timeout of some firewall is too long, which cause the host easy to be crashed. So, shortening TCP idle timeout will decrease the opportunity to fill session table [5]. But if timeouts are too short, session table will be inserted and deleted frequently.

Processing session timeout is essential for connection-level monitoring devices such as Stateful Inspection firewalls in order to minimize security holes. Adequate session timeouts can improve the security of both protected host and firewall itself.

This paper proposed a new architecture for session processing, which improves the timeout processing performance of session table effectively, and securities of both protected hosts and firewall itself.

## 2 Session Table Processing

Generally, <src-addr, src-port, dst-addr, dst-port, ip-> is used to identify a unique session, which is called SID in this paper. For each arriving packets, session table is looked up for a match. If packets belong to an existing session, session state and session time will be updated. If a session entry has overtime, it will be deleted.

Generally, there are 4 kinds of session table operation: match session table with packets' sessionID, update an entry's time and state, insert a new entry and delete an overtime entry, which are shown in Fig.1. Because an update always is after a match, we call them a match-and-update operation.

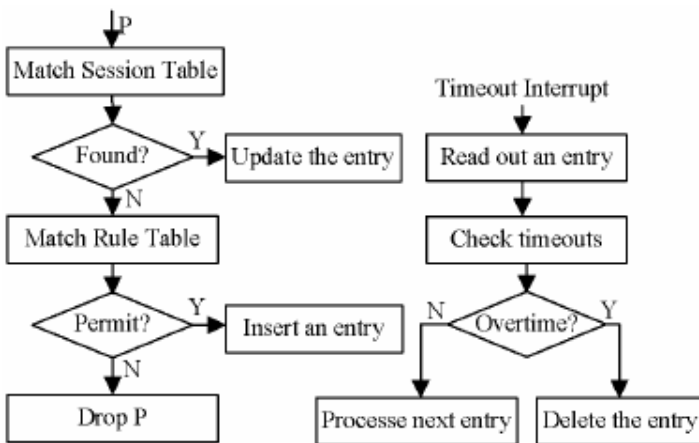


Fig. 1. Session table operations

Required time for inserting an entry ( $T_{ins}$ ), deleting an entry ( $T_{del}$ ), and match-and-update( $T_{mau}$ ) are three key parameters to the performance of Stateful Inspection firewalls. Because memory access is most time-consuming, we use the required clock cycles of memory accesses to measure  $T_{ins}$ ,  $T_{del}$  and  $T_{mau}$ .

### 2.1 Existent Processing Methods of Session Table

Now all existed firewalls put both SID and <state, time> in a single entry, as Fig.2. Because the number of entries may up to 1 million, and each entry is wider than 128 bits, over 128Mb memory space is required for session table. Generally, DDR SDRAM is used to store session table.

Src_addr	dst_addr	Src_port	dst_port	protocol	State	Time
----------	----------	----------	----------	----------	-------	------

Fig. 2. General format of session table entry

Because SID is about 128 bits long, comparing a packet SID with session table entries requires an efficient searching algorithm that performs searching without comparing the whole 128 SID bits. One such efficient searching algorithm is PATRICIA. PATRICIA is an algorithm that needn't compare the whole key. PATRICIA is very useful for extremely long key. PATRICIA trie compress all nodes which have one-way branch, it is the lowest trie [7]. But traversing PATRICIA trie to process timeout is time-consuming. In order to improve traversing performance, leaves should be linked to a rope. Fig.3 illustrates the data structure in [8] for fixed-length match, and Fig.4 illustrates the Leaf Rope of PATRICIA trie which is a single linked list. All leaves in the PATRICIA trie are linked as a rope in inserting time order.

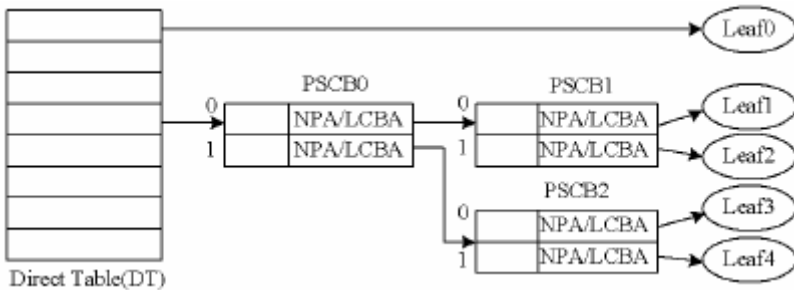


Fig. 3. Data Structure of IBM NP4GS3's FM [8]

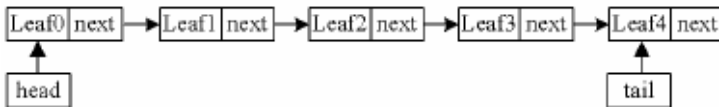


Fig. 4. Leaf Rope structure

$T_{search}$ ,  $T_{insert}$  and  $T_{delete}$  are respectively required time of searching, inserting and deleting an entry in PATRICIA trie. From the processing in Fig.1 and PATRICIA algorithm in [9], we get the following functions, which have considered the time cost

of applying and reclaiming memory block. The  $R_{ot}$  is the ratio of the number of overtime entries to all entries, and  $T_{DDR}$  is required clock cycles of accessing DDR memory.

$$T_{mau(1)} = T_{search} + T_{DDR}$$

$$T_{ins(1)} = T_{insert} + 2T_{DDR}$$

$$T_{del(1)} = T_{delete} + T_{DDR} / R_{ot}$$

$R_{ot}$  generally is very small, which is far less than 1, so timeout processing is time-consuming. When processing many overtime entries once, the time cost of timeout processing is intolerable.

### 2.2 Doubly Linked List Structure

From the timeouts processing of Fig.1, we know that to improve timeout processing performance, the rope should be linked in updating time order. If the rope is linked as a doubly linked list in time order, we need not read many entries when processing timeout. When an entry does not timeout, all subsequent entries do not timeout too. Fig.5 shows the data structure.

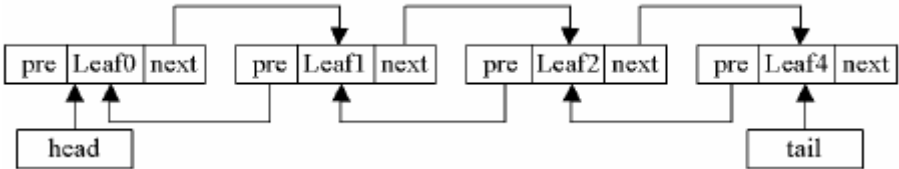


Fig. 5. PATRICIA based doubly linked list

$$T_{mau(2)} = T_{search} + 7T_{DDR}$$

$$T_{ins(2)} = T_{insert} + 2T_{DDR}$$

$$T_{del(2)} = T_{delete} + T_{DDR}$$

This architecture can improve the performance of timeout processing effectively, but it causes the performance of match-and-update down heavily. The main reason is that it needs too many memory accesses when updating session table. It needs 7 memory accesses at least, and DDR is very slow.

### 2.3 Proposed Architecture

Higher speed memory than DDR can improve the performance of session table. In this paper, we proposed a new architecture for session table. We use two kinds of memory to store session table, which can improve the performance of session table greatly. Because SID is wide and session table generally have too many entries, we uses DDR SDRAM to store SIDs PATRICIA trie.

In order to decrease  $T_{del(1)}$ , we use a doubly linked list to organize <state, time>. Fig.6 shows its data structure. ZBT SRAM is used to store the doubly linked list. We use  $DS1\_addr$  and  $DS2\_addr$  to relate the two data structures, as Fig.7.

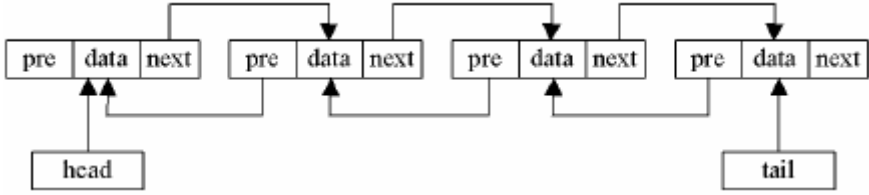


Fig. 6. Doubly Linked Lists for DS2

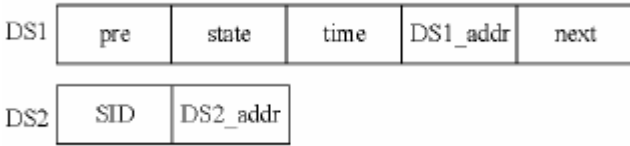


Fig. 7. Entries of DS1 and DS2

Algorithm 1. Insert a session entry.

- (1) apply a DDR memory block for DS1 and a ZBT memory block for DS2.
- (2) fill the DS1 block with sessionID and the DS2 block’s address, and insert it in PATRICIA trie.
- (3) fill the DS2 block with state, time and the DS1 block’s address, insert it in doubly linked list.

Algorithm 2. Match and update a session entry.

- (1) search PATRICIA trie for a match, if matched use DS2\_addr as address to read DS2 node.
- (2) update the DS2 node with new state and current time.
- (3) delete the DS2 node from its old position.
- (4) insert the DS2 node in the tail of list, update list’s tail.

Algorithm 3. Timeout processing.

- (1) when received a timer interrupt, read out the first DS2 node from its head.
- (2) compare the time in DS2 with current time to determine whether it is overtime or not.
- (3) if overtime delete the DS2 node, update list’s head, and delete the corresponding DS1 node.
- (4) process its next node.

Following functions measure the performance parameters of the proposed architecture.

$$T_{mau(3)} = T_{search} + 7T_{ZBT}$$

$$T_{ins(3)} = T_{insert} + 3T_{ZBT}$$

$$T_{del(3)} = T_{delete} + 2T_{ZBT}$$

## 2.4 Pipeline the Proposed Architecture

$T_{search}$ ,  $T_{insert}$  and  $T_{delete}$  only involve DDR SDRAM. Because  $T_{DDR}$  needs about 16 cycles for 32 bits memory data bus and 256 bits data,  $T_{search}$ ,  $T_{insert}$  and  $T_{delete}$  at least need 32 cycles.  $T_{ZBT}$  only involves ZBT SRAM, and  $T_{ZBT}$  only need 2 cycles for 36 bits memory data bus and 72 bits data. We get  $7T_{ZBT} < T_{search}$ ,  $3T_{ZBT} < T_{insert}$  and  $2T_{ZBT} < T_{delete}$ . If we pipeline DS1 operation and DS2 operation, the performance of session table will only be determined by  $T_{search}$ ,  $T_{insert}$  and  $T_{delete}$ . Following functions show the performance parameters of the pipelined architecture.

$$\begin{aligned} T_{mau(4)} &= T_{search} \\ T_{ins(4)} &= T_{insert} \\ T_{del(4)} &= T_{delete} \end{aligned}$$

## 2.5 Performance Analysis

In our design,  $T_{DDR}$  equals 16 cycles and  $T_{ZBT}$  equals to 2 cycles. From above analyses, we know that  $T_{del(4)}$  is much less than  $T_{del(1)}$ , and  $T_{mau(4)}$  is much less than  $T_{mau(2)}$ . We can know that the pipelined architecture is the best architecture.

From above analyses, we know that  $T_{search}$ ,  $T_{insert}$  and  $T_{delete}$  determine session table's performance. So to minimize  $T_{search}$ ,  $T_{insert}$  and  $T_{delete}$  is the most important thing to improve the performance of session table processing. On the base of traditional PATRICIA trie we proposed a new PATRICIA trie, which improves the performance of traditional PATRICIA insertion [9].

# 3 Dynamical Timeouts

## 3.1 Queue Structure

An important advantage of the new architecture is that we can easily organize multiple double linked lists in ZBT SRAM, and each double linked list is called a queue in this paper. We can set dynamical timeouts for each queue, and each queue's timeouts can be different, which can improve securities of both protected hosts and firewall itself.

To illustrate our methods for dynamical timeouts, we use typical topology of network as example, as Fig.8. In this paper, we assume the HTTP serve and FTP serve need be protected against SYN Flood attack, and firewall itself need defense other TCP attack.

We use ZBT SRAM to store DS2 data structure. We design four queues for different functions. Queue 1 and Queue 2 are respectively used to store session tables which have received the SYN/ACK from the HTTP server and FTP server. Queue 3 is used to store session tables which have established TCP connection but not receive any FIN or RST packets. Queue 4 is used to store all others' session tables. Fig.9 shows queues and sessions flow examples.

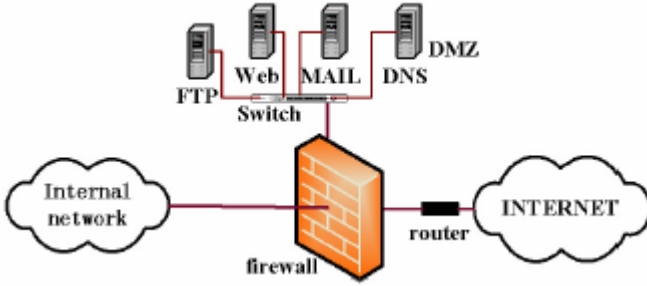


Fig. 8. Typical firewall topology

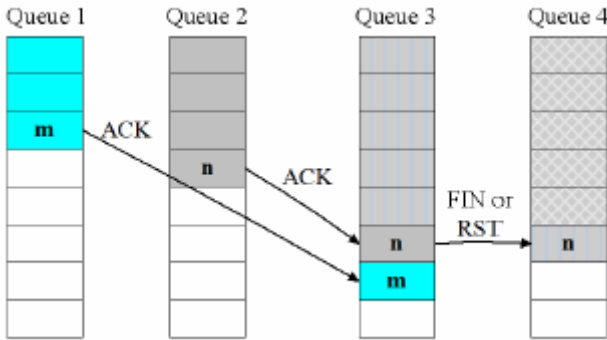


Fig. 9. Four queues used for storing sessions

### 3.2 Algorithms for Queues

For each arriving packet, the PATRICIA Trie is looked up for a match.

Algorithm 6: SYN packets processing.

- (1) if find a session entry, drop the packet, else lookup in the rule base for a match.
- (2) if does not be allowed, drop it, else use Algorithm 1 to insert the new entry in Queue 1.

Algorithm 7: SYN/ACK packets processing.

- (1) if not find a session or the session's state is wrong, drop the packet.
- (2) else if destination IP is HTTP server, move the session table entry from Queue 4 to Queue 1, else if destination IP is FTP server, move the session table entry from Queue 4 to Queue 2, else move the session table entry to the tail of Queue 4.

Algorithm 8: ACK packets processing.

- (1) if not find a session or the session's state is wrong, drop the packet.
- (2) else move the session table entry to the tail of Queue 3.

Algorithm 9: packets with FIN or RST.

- (1) if not finding a session or the session's state is wrong, drop the packet.
- (2) else move the session to the tail of Queue 4.

### 3.3 Dynamical Timeouts

For the above four queues, the first two queues is mainly related with the security of protected host, and the last two queue is mainly related with the security of firewall itself. If defining  $T$  is the maximum of timeout,  $N$  is the whole resource, and  $M$  is available resource, we proposed a new timeout model. Fig.10 illustrates timeouts curve with  $M$  varying.

$$\text{Timeout} = T \frac{1}{2^{\lfloor \lg \frac{N}{M} \rfloor}}$$

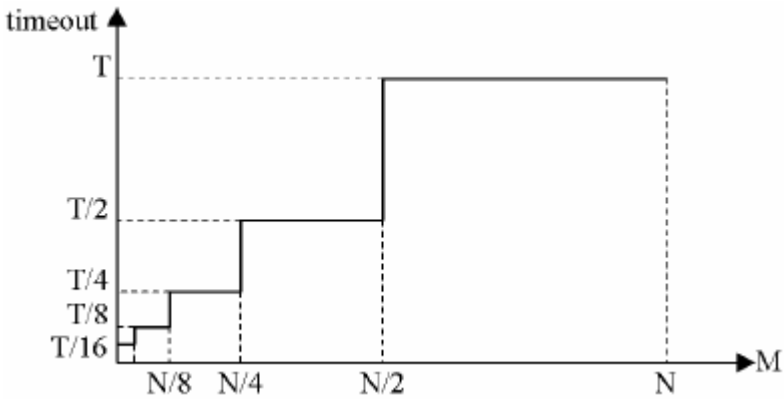


Fig. 10. Timeouts curve

For timeouts of the first two queues, selecting  $T$  is 32 seconds,  $N$  is configured from user which typically is the number of half open host supported, and  $M$  is available resource in protected host. To insure the slow network user can through firewall normally, minimal timeouts of Queue 1 and Queue 2 are 1 second. The  $N$  and  $T$  may different for the two queues according to users' configuration.

For timeouts of Queue 3, selecting  $T$  is 4096 seconds,  $N$  equal to the whole space of ZBT memory, and  $M$  is available resource in ZBT SRAM. The minimum of timeout is 256 seconds.

For timeouts of Queue 4, selecting  $T$  is 32 seconds,  $N$  equal to the whole space of ZBT memory, and  $M$  is available resource in ZBT SRAM. To insure the slow network user can through firewall normally, its minimum of timeout is 1 second.

When timeouts of any queue has decreased to its minimum, each new connection will purge an oldest connection in the session table.

### 3.4 Performances and Securitys Analyses

Dynamical timeouts improve securitys of both protected host and firewall itself.

(1) when processing timeout as Algorithm 3, only access part of table entries because the linked list is in the order of updating time, which improves the performance



of timeout processing. By pipelining, the time cost of timeout processing is equal to the time cost of deleting the DS1 entry.

(2) when updating session table entry as Algorithm 2, By pipelining, the time cost of updating session is equal to the time cost of matching a DS1 entry.

(3) as Algorithm 2, 6 and 7, firewall uses special queues for protected hosts to defense SYN flood, and sets timeouts of Queue 1 and Queue 2 according to available resource of the protected host, which can improve protected hosts' security against SYN Flood. Furthermore, when timeouts of Queue 1 and Queue 2 down to the minimum, each new inserted entry will cause the oldest entry purged, which can insure the protected hosts not crashed.

(4) as Algorithm 2, 7, 8 and 9, firewall sets timeouts of Queue 3 and Queue 4 according to available resource of firewall itself, which can improve both protected hosts and firewalls' security. Furthermore, when timeouts of Queue 3 or Queue 4 down to the minimum, each new inserted entry will cause the oldest entry purged, which can insure the firewall not crashed.

## 4 Experimental Design and Performance Analyses

For high speed network, ASIC is often used to improve performance of network device. We implement an ASIC for the proposed session table architecture and proposed algorithms. We use two methods to reduce the depth of PAT-FM trie. One is to hash SID, the other is to make use of 4-ary PATRICIA trie. We use a Xilinx FPGA(XC2V3000) to implement the whole Stateful Inspection firewall which supports 3 Gigabit Ethernet ports and a PCI interface. PATRICIA trie is stored in DDR SDRAM, doubly linked list is stored in ZBT SRAM.

We use random 128 bits data to test the performance of session table insertion, search and deletion. Table 1 shows the experimental results. We can know that performances of session entries insertion, search and deletion are very close. All of these performances are mainly determined by PATRICIA's performance.

Furthermore, we mainly test the performance of lookup session table for a match, because lookup performance is the most important parameter for Stateful Inspection firewalls. Fig.11 shows the experimental result, the ASIC for the new architecture and new algorithms can do 2.78 million lookups even the number of session entries up to 1 million. If all packets are smallest packet (64 bytes), the application-specific hardware can process 1.7 Gbps' traffic. And when the average size of packets is 128 bytes, the traffic can up to 3.4 Gbps. So, we can know that this ASIC can work well in the Gigabit edge network device.

**Table 1.** Operation numbers per second

Search	Insertion	Deletion
2,464,274	2,796,528	2,471,232

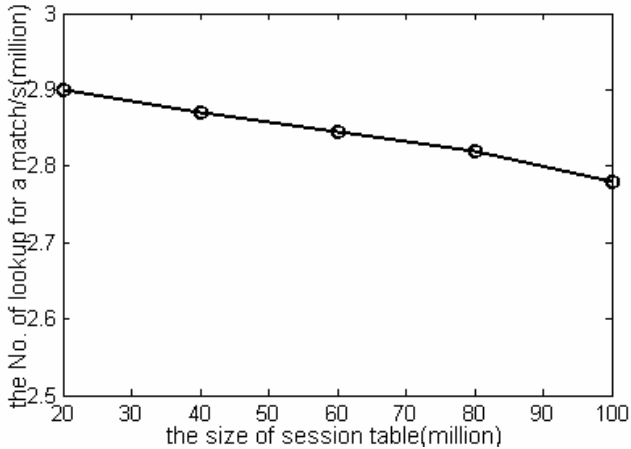


Fig. 11. Lookup performance of session table

## 5 Conclusions

The Stateful Inspection technology is a key to network firewall, and the performance of session table is a key to the Stateful Inspection technology. The new architecture improves performances of both timeout processing and session updating. Dividing session entry into two separate parts and designing different data structures for each other improves the performance of session table. By pipelining operations of the two parts, the performance of session table is only determined by PATRICIA trie's performance. The new fixed-length match PATRICIA algorithm can improve insertion performance effectively.

Because the new architecture sets dynamical timeouts according to available resource of both protected host and firewall, which insure both protected hosts and firewall itself not SYN-Flooded. The specialized hardware for the proposed architecture is implemented in FPGA, which improve firewall's performance further. By analyzing the architecture and experimental results, the new session table architecture can work well in Gigabit Ethernet network.

## References

1. Stateful-inspection firewall: The Netscreen way. [http://www.netscreen.com/products/firewall\\_wpaper.html](http://www.netscreen.com/products/firewall_wpaper.html).
2. David, W., Chapman, Jr., Andy, F.: Cisco Secure PIX Firewalls, Cisco Press (2001)
3. [http://www.3com.com/other/pdfs/products/en\\_US/400742.pdf](http://www.3com.com/other/pdfs/products/en_US/400742.pdf).
4. Marcus, G., Steven, B.: Check Point Firewall-1 Administration Guide (2001)
5. Inhye, Kang, Hyogon, Kim: Determining embryonic connection timeout in Stateful Inspection, IEEE 2003 International Conference on Communication. Anchorage, USA (2003) 458-462

6. Noureldien, A.N.: Protecting Web Servers from DoS/DDoS Flooding Attacks, International Conference on Web-Management for International Organization. Geneva (2002)
7. Okuno, M., Ando, K., Aoe, J.: An efficient compression method for Patricia tries, IEEE International Conference on Computational Cybernetics and Simulation, Vol.1 (1997) 415–420
8. IBM Co.: IBM NP4GS3 DATAsheet (2001)
9. Li, X, Hu, M.Z, Ji, Z.Z, A Hardware-Based PATRICIA Algorithm for Fixed-length Match, Computer Research and Development (2005) 951–957

# A Behavior-Based Ingress Rate-Limiting Mechanism Against DoS/DDoS Attacks\*

Song Huang, Ling Zhang, and Shou-Ling Dong

Guangdong Key Laboratory of Computer Network,  
South China Univ. of Tech. , GZ, P.R. China 510641  
{crshuang, ling, sldong}@scut.edu.cn

**Abstract.** In this paper, the characteristics of Client/Server interaction behaviors under normal web access and typical DoS/DDoS attack are analyzed. A simple local rate-limiting method called Behavior-based Ingress Rate-limiting (BIR) mechanism is proposed, by which the client-end host's inbound and outbound traffics are monitored. Bursts of the traffics are suppressed by a local transmission delay mechanism. The principle and implementation are described. Simulations are performed to validate its efficacy. Finally, the approach's potential and limitations are also discussed.

## 1 Introduction

DoS/DDoS attacks are easy to launch but hard to prevent. Generally, they are caused by the imbalance among the Internet between the clients and suppliers. Countermeasures against DoS/DDoS attacks can be approximately classified into several categories: a) filtering or rate-limiting mechanisms to decrease the number of requests; b) increasing the capability of service provider; c) network structure optimization, including upgrade the architect of the Internet, or improve related protocols.

The request filtering or rate-limiting mechanisms, according to their locations, can be ascribed to victim-end, intermediate-network based and source-end. Source-end mechanisms have relatively higher efficiency, but not easy to deploy and control.

DDoS attack is the distributed form of DoS one, which is launched by employing several masters and many agents. Masters refer to compromised hosts to command more agents. They receive instructions of the attacker, and retransmit to downstream agents. Agents receive instructions from masters, and send flooding packets directly to the victim. Here, the agents act as the sources of the attacking traffics.

Inspired by the simple technique to suppress the fast spread of viruses[1], a similar approach against DoS/DDoS attacks is proposed in this paper. Its principle is to prevent Dos/DDoS attacks by rate-limiting burst requests of each individual host. This simple approach is based on attacker's behaviors, and may acts as a complementary mechanism of existing defending system.

---

\* This research was sponsored by National "973" plan(2003CB314805).

The paper is organized as follows. Section 2 discusses the behavioral patterns based on experiments. A detailed description on BIR mechanism is given in section 3. Section 4 illustrates the efficacy validation by simulations. Section 5 provides a brief description of related works. Section 6 concludes the paper, with some discussion of its potential and limitations.

## 2 Behavior Patterns

Matthew M. Williamson [1] proposed an approach to restrict virus’s high speed propagation automatically, which is based on the observation that during virus propagation, an infected machine will connect to as many different machines as fast as possible. The approach’s principle is to limit the rate of connections to “new” machines by both slowing and halting this propagation without affecting normal traffic.

With the similar principle, BIR is designed to limit the rate of requests to a specific machine. This mechanism is deployed on local source-end host, and keeps monitoring the local host’s traffic anomaly. Then it performs predefined rate-limiting actions once needed.

Obviously, the key of this design is to find the common characteristics of agents sending flooding traffics or are suspected to do so. Inspired by Matthew Williamson’s throttling method, we wish to find a representative behavior of the attacking traffics. A simple experimental environment is setup to collect traffics data for typical Internet services, including web browsing, web mailing and file transferring. Only traffics above TCP/UDP layer are considered.

### 2.1 Normal Situations

Sniffer Pro® is used to collect traffics of web browsing services.

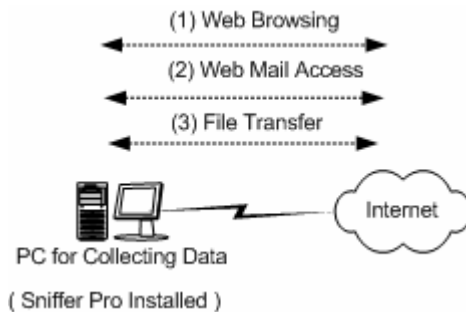
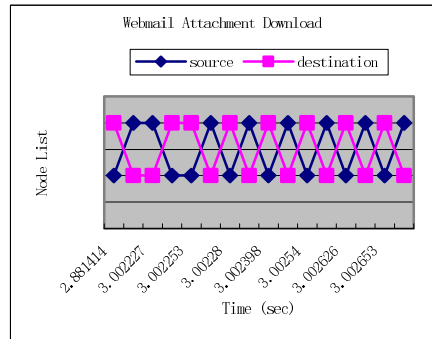
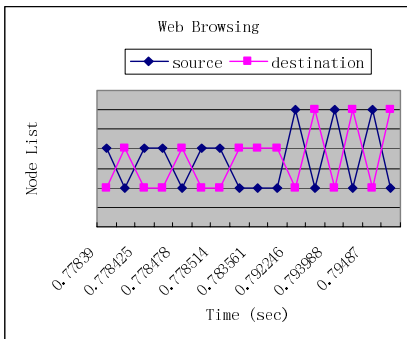


Fig. 1. Data collection experiment

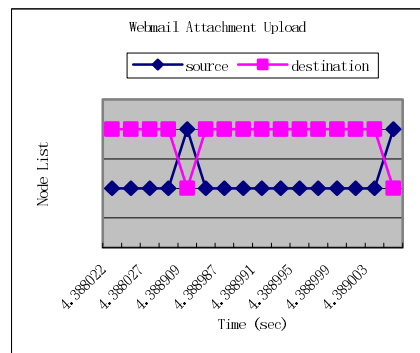
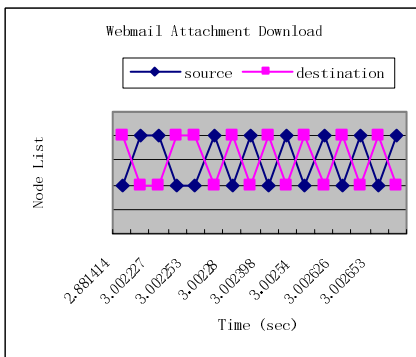
The client’s inbound and outbound traffics are saved and plotted for analysis. Three typical network services are studied: (1) web browsing; (2) web mailing; (3) file transfer. Part of the data is listed in Table.1, and the plot is shown in fig 2. The client’s address is 222.16.32.96. The servers’ addresses are 202.205.3.130 and 202.205.3.140.

**Table 1.** Interaction Record(segment) of Web Browsing

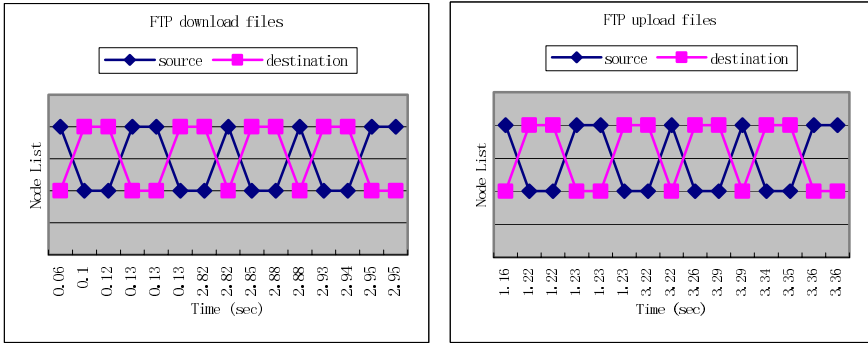
Time	Source	Destination
0.77839	202.205.3.130	222.16.32.96
0.778411	222.16.32.96	202.205.3.130
0.778425	202.205.3.130	222.16.32.96
0.77846	202.205.3.130	222.16.32.96
0.778478	222.16.32.96	202.205.3.130
0.778492	202.205.3.130	222.16.32.96
0.778514	202.205.3.130	222.16.32.96
0.77853	222.16.32.96	202.205.3.130
0.783561	222.16.32.96	202.205.3.130
0.78474	222.16.32.96	202.205.3.130
0.792246	202.205.3.140	222.16.32.96
0.79229	222.16.32.96	202.205.3.140
0.793988	202.205.3.140	222.16.32.96
0.794031	222.16.32.96	202.205.3.140
0.79487	202.205.3.140	222.16.32.96
0.794914	222.16.32.96	202.205.3.140



**Fig. 2.** Source-destination interactions under web browsing scenario



**Fig. 3.** Source-destination interactions under web mailing scenario



**Fig. 4.** Source-destination interactions under file transferring scenario

From these figures, some common characteristics can be derived: the interactions between a client and a server are fairly balanced. Statistically, the ratio of outbound traffics to inbound ones of a specific connection approximately remains constant for each scenario. For normal web browsing service, the ratio is about 1:1. For web mail with large attachment transferring, the ratio increases up to about 1: 10.

**2.2 Under Attacks**

During a flooding attack, the client keeps sending requests without caring about the destination’s responses. This makes the ratio of the outbound to inbound packets increase rapidly. For cases of source address spoofing, even fewer responses can reach the source side.

**Table 2.** Source-destination communication during a Syn-flood

Time(sec)	Source	Destination
0.000075	201.79.131.49	202.38.192.2
0.000094	201.79.131.50	202.38.192.2
0.000113	201.79.131.51	202.38.192.2
0.000132	201.79.131.52	202.38.192.2
0.000151	201.79.131.53	202.38.192.2
0.000169	201.79.131.54	202.38.192.2
0.000188	201.79.131.55	202.38.192.2
0.000207	201.79.131.56	202.38.192.2
0.000226	201.79.131.57	202.38.192.2
0.000245	201.79.131.58	202.38.192.2
0.000264	201.79.131.59	202.38.192.2
0.000283	201.79.131.60	202.38.192.2
0.000302	201.79.131.61	202.38.192.2
0.000321	201.79.131.62	202.38.192.2
0.00034	201.79.131.63	202.38.192.2

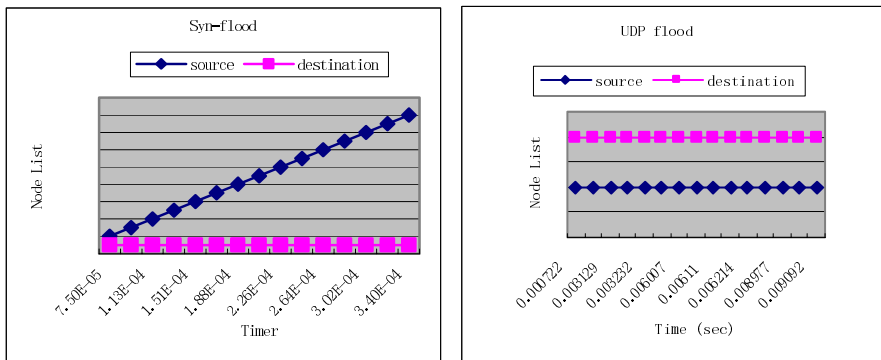
**Table 3.** Source-destination during a UDP-flood

Time(sec)	Source	Destination
0.000722	222.16.32.96	222.16.32.126
0.003115	222.16.32.96	222.16.32.126
0.003129	222.16.32.96	222.16.32.126
0.003225	222.16.32.96	222.16.32.126
0.003232	222.16.32.96	222.16.32.126
0.00598	222.16.32.96	222.16.32.126
0.006007	222.16.32.96	222.16.32.126
0.006099	222.16.32.96	222.16.32.126
0.00611	222.16.32.96	222.16.32.126
0.006207	222.16.32.96	222.16.32.126
0.006214	222.16.32.96	222.16.32.126
0.008961	222.16.32.96	222.16.32.126
0.008977	222.16.32.96	222.16.32.126
0.009077	222.16.32.96	222.16.32.126
0.009092	222.16.32.96	222.16.32.126

Here, a SYN-flood attack with address spoofing and a UDP-flood attack are simulated in the experimental environment. Interactions between the client and the server are recorded, and part of which are shown above.

### 2.3 Result Analysis

The interactions between the client and the server are quite different under normal or attack scenarios. This is caused by different incentives and behaviors of ordinary users and attackers. An ordinary user accesses services in an interactive way. When the response is slow or delayed, he may tend to slow down his requests, or turns to an alternative website. But an attacker has a different incentive. High intensity requests



**Fig. 5.** Source-destination interactions during Syn-flood and UDP-flood



flows are sent to the victim, caring nothing about whether the target is busy, or whether the link is in congestion. This behavior leads to an inconstant imbalance between inbound and outbound traffics, and it may become even worse with source address spoofing

### 3 Behavior-Based Ingress Rate-Limiting

A Behavior-based Ingress Rate-Limiting (BIR) approach is proposed to suppress flooding requests from the beginning.

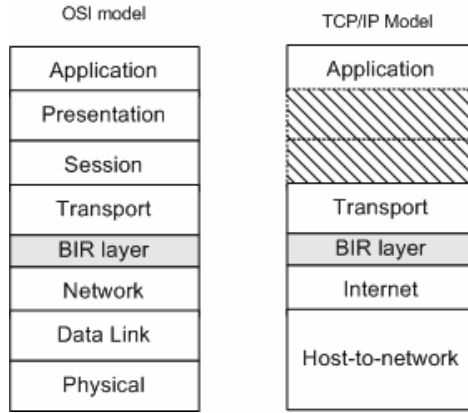


Fig. 6. BIR layer in protocol stack

The principle is based on following assumption: If the difference between inbound and outbound packets of a client-end host remains at approximately constant, it represents a normal situation. While the difference keeps increasing in a dramatic high rate, the host may be launching flooding attacks. Then countermeasures should be taken to suppress suspicious outgoing traffics by delaying the outbound packets. In this way, a burst flooding requests flow can be flatted to a moderate one.

In BIR, the policy of delaying rather than packet dropping is adopted to avoid possible negative impacts on legitimate traffics.

BIR can be realized as a sub-layer between TCP/UDP and IP layers in network protocol stack, performing monitoring and delaying operations.

#### 3.1 Traffic Monitoring

To monitor the host’s traffic, new data structure need be defined at first, which includes:

the suspicious traffics list

$$Ls\{(A_d, I_d, P_d, N_s), N_{Max}, T_C\} \tag{1}$$

the outbound delaying queue

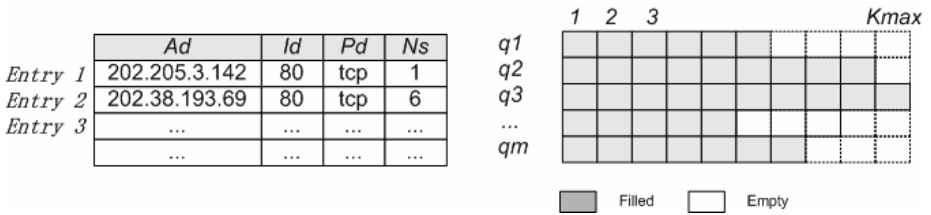
$$Q_s \{q[n] (n = 1, 2, \dots, K_{Max}), T_d \} \tag{2}$$

$A_d$  -- destination address,

$I_d$  -- port number

$P_d$  -- protocol type;

$N_s$  -- number difference between outbound and inbound packets.  $N_s > 0$  means the outbound packets is more than the inbound ones;



**Fig. 7.** Data structure illustration

$N_{Max}$  -- upper limit of out / in traffics difference.  $N_s > N_{Max}$  representing the difference exceeds the normal limit, then following outbound packets to the same IP address and port number will be send to  $Q_s$  for outbound traffic delaying.

$T_C$  -- updating interval of  $L_s$ ; Each time a  $T_C$  is running out, the  $L_s$  will be refreshed by clearing entries that need not being monitored any longer.

$q[n]$  -- subqueue of  $Q_s$ , and stores the outbound packets for one specific target address. The delay time range from  $T_d$  to  $K_{Max} \times T_d$ .

$K_{Max}$  -- maximum length of  $q[n]$ .

$T_d$  -- basic time delay unit.

The mechanism in pseudo-code is as follows :

1. Once sending a packet, the dest-address, port number, together with the protocol type are extracted, then matched with  $(A_d, I_d, P_d)$  entries in  $L_s$ .
  - 1.1. If the match fails, the packet's parameters will be put into  $L_s$  as a new entry, and its  $N_s$  value increases by one.
  - 1.2. If match succeeds, its  $N_s$  added by one, then:
    - If  $N_s > N_{Max}$ , the packet is inserted into  $Q_s$  for a delay;
    - If  $N_s \leq N_{Max}$ , the packet is sent out directly;

2. Once receiving a packet, the source address, port number, and protocol type will be extracted, then matched with each entry. If succeed, the entry's  $N_S$  value will be decreased by one.

In order to eliminate accumulation effect,  $L_S$  is checked periodically, and those entries'  $N_S$  values will be set to zero if their  $N_S \leq N_{Max}$ .

BIR only monitors a fixed number of communications. To prevent  $L_S$  from overflowing, once  $L_S$  is full, new entries will replace the old ones. For example, entries with  $N_S = 0$  can be replaced firstly.

### 3.2 Outbound Delay

All suspicious outbound packets are put into  $Q_S$  for a delay. The delay time for each packet in  $L_S$  is a function of  $(N_S - N_{Max})$ . The more difference between  $N_S$  and  $N_{Max}$ , the more likely the outbound packets belong to flooding traffics, then the longer delay should be performed. The relationship of the time delay and  $(N_S - N_{Max})$  can be a monotonous increasing functions such as a linear function or an exponential one, according to the actual algorithm's implementation.

$Q_S$  will be scanned every  $T_d$  time, to send packets running up their delay time.

Packets in  $Q_S$  are sent out every  $T_d$  time, so the highest transmitting rate is:

$$R_{Allowed} = 1/T_d \quad (3)$$

delay time is:

$$t_{Delay} = f(N_S - N_{Max}) \times T_d \quad (4)$$

$f()$  is a predefined monotonously increasing function. For a linear increasing function with proportion of  $K_d$ , we have  $f(x) = K_d x$ , where  $K_d$  represents the sensitivity to the outbound-inbound difference  $(N_S - N_{Max})$ .

As to normal interactions, the value of  $(N_S - N_{Max})$  remain a low level, therefore the delayed time is zero. For attacking traffics with real source addresses, the  $N_S$  increases rapidly until bigger than  $N_{Max}$ , which make following outbound packets be put into  $Q_S$ . For attacking traffics with forged source addresses, fewer responses returned from the victim makes the increase of  $N_S$  even more rapidly, which in turn actuates the rate-limiting mechanism more rapidly.

### 4 Simulation

Simulations are carried out using OPNET® software. The network topology is shown in Fig.8. N1 to N12 denote clients. s1 and s3 denote two switches. r1 and r2 denote two routers. Server is the destination of all flooding traffics, which provides all services.

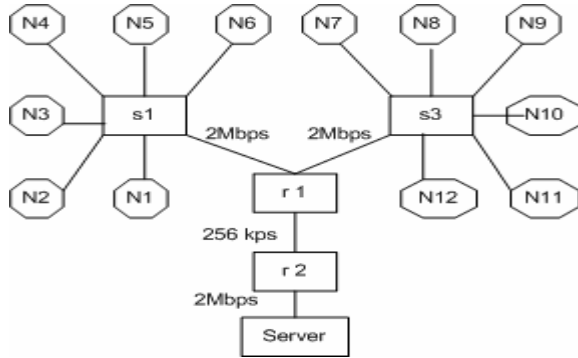


Fig. 8. Network’s Topology of the Simulation

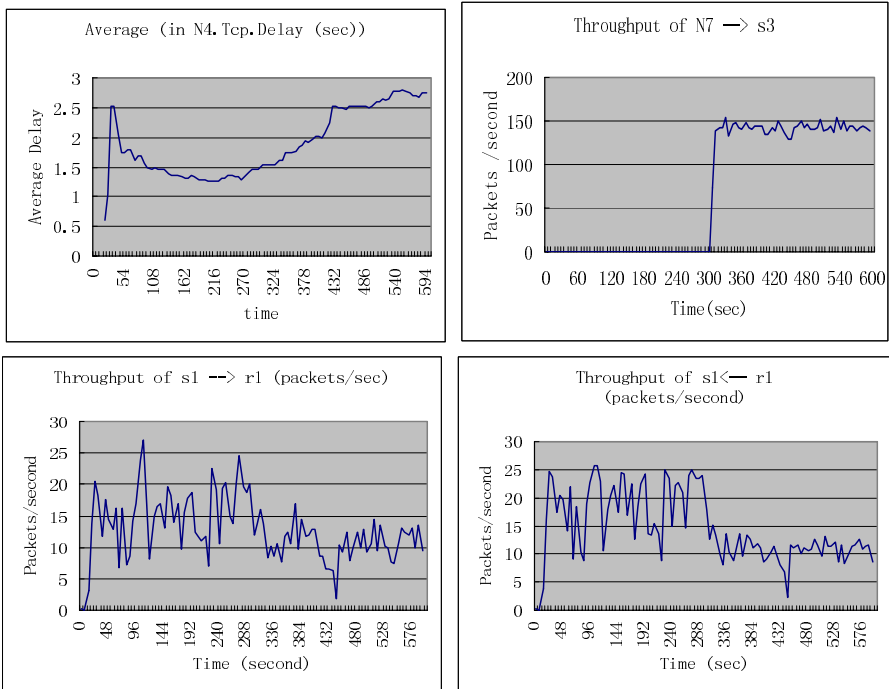


Fig. 9. Flooding attack

In order to distinguish traffics clearly, we perform normal web accessing services on nodes N1-N6, and perform sudden flooding requests on nodes N7-N12. Therefore, the traffics of link s1-r1 represent normal web browsing, while traffics of link s3-r1 represent flooding requests. In this way, we can watch the influence of flooding traffics clearly.

The bandwidth of link r1- r2 is set as 256kbps, which far less than others. This link is used to simulate the bottle neck of network links under DoS/DDoS flooding attacks.

### 4.1 Under Attacks

First, we simulate the situation under flooding attacks. The overall simulation last 600 seconds, and four nodes selected from N7-N12 launch TCP flooding requests at moment of the 300<sup>th</sup> second.

In Fig.9, Flooding requests to the server are launched at the 300<sup>th</sup> second. Before the flooding attacks, TCP delay of N4 comes to almost a stable state around 1.3. Under attack, TCP delay of N4 keeps increasing to 2.7 seconds, which means slower responses of web browsing under flooding attacks. The throughput of the link s1-r1 drops significantly, which is mainly caused by bottle neck link between r1 and r2.

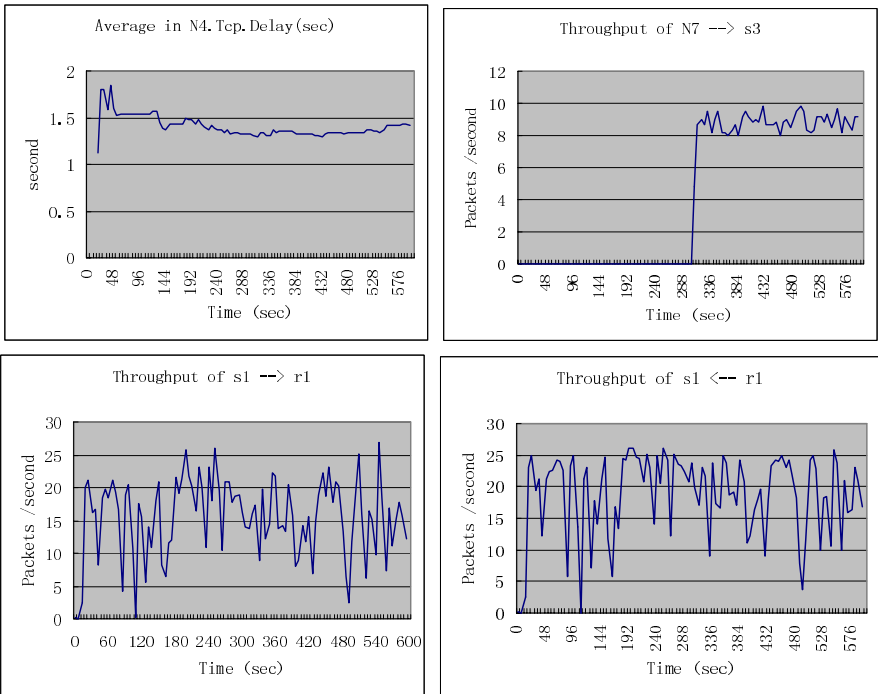


Fig. 10. Flooding attack with BIR

## 4.2 Suppress the Burst with BIR

With BIR mechanisms deployed on N1-N12, the situation is improved significantly. When flooding attacks are launched at the 300<sup>th</sup> second, the traffic of link s1-r1 remains almost the same stable level. And the throughput uprising of link N7 to s3 is successfully suppressed to only around 9 packets /second, which is almost one fifth of the previous uprising magnitude.

## 5 Related Works

Quite a few source-end filtering or rate-limiting defending schemes have been developed to suppress flooding attacks from the beginning.

Haining Wang[2] proposes a simple and robust mechanism for detecting SYN flooding attacks. The approach detects the SYN flooding attacks at leaf routers that connect end hosts to the Internet. The principle is based on the protocol behavior of TCP SYN-FIN(RST) pairs. The efficacy of this detection mechanism is validated by trace-driven simulations. However, as stated in this paper, once the attacker is aware of the presence of such a detection system, it can paralyze the SYN-FIN detection mechanism by flooding a mixture of SYNs and FINs(RSTs).

In literature [3], a simple and robust approach called D-SAT system is proposed to detect SYN flooding attacks by observing network traffic. Instead of managing all ongoing traffic on the network, D-SAT only monitors SYN count and ratio between SYN and other TCP packets at first stage. And it detects SYN flooding and finds victims more accurately in its second stage. The simulation demonstrate that D-SAT system is efficient and simple to implement and prove that it detects SYN flooding accurately and finds attack in a very short detection time.

Compared with SYN-FIN(RST) pair or D-SAT, BIR mechanism locates at individual host rather than leaf routers. A comparatively simple rate-limiting method is adopted by BIR, while the detection is not so accurate as that of D-SAT. However, without the need to handle multiple traffic flows at leaf routers, less resource and computation capability are required.

D-WORD[4][5] is a well known source-end DDoS defense system. D-WARD detect attacks by the constant monitoring of two-way traffic flows between the network and the rest of the Internet, and periodic comparison with normal flow models. Mismatching flows are rate-limiting in proportion to their aggressiveness.

BIR has a similar principle as D-WARD, except that BIR is realized at host, while D-WARD at source router. If the source router deploying D-WARD is not the only border router of the source network, it might not see both direction of the flow for certain peers, and thus will not classify these flows properly.

Generally, most source-end detection systems are actually Intrusion Detection Systems (IDS), which require at least three main parts deployed across the overall networks: detector, analyzer and actuator. All these parts basically form a self-regulating reverse-feedback system. Firstly, their detection and control algorithm are usually complex and accurate. Most of them are usually implemented at routers, which may bring negative influence to routers' performance. Furthermore, the collaboration of all distributed components across the network is also a challenging problem.

BIR is comparatively a kind of simple and light overhead module which is deployed at individual host. BIR can be deployed as a software patch. Hosts are not required to collaborate with each other, although it may be possible for source-end hosts to communicate with the router or the victim in the future. Comparatively, the BIR is not as accurate as other approaches. In essence, it is a behavior self-restraining mechanism for hosts of the Internet.

Being different from TCP's congestion control, which is a reactive mechanism, BIR is a preventive one. In addition, it detects and reacts in a faster way.

The great complexity of the DDoS problem suggests that its solution will require the use of multiple defenses, such as rate-limiting, filtering, trace back, push back, and so forth. BIR is appropriate to be a component in such an integrated defense system.

## 6 Conclusion

BIR can be deployed by stages. As a behavior self-restraining mechanism, BIR requires little coordination among hosts and domains. The more hosts equipped with BIR, the more effects can be achieved.

There are some limitations for BIR. It is not so effective while facing extremely distributed DDoS attacks, during which each agent generates a moderate traffic while the aggregate of them can still overwhelm the victim. Another problem occurs if attacker sends ACKs or other response packets to each agent periodically to bypass BIR monitoring mechanism. Besides these problems, services analyzed in this paper are all ordinary ones as web browsing, web mailing, file transfer. While dedicated network applications may have different interaction model, which still needs further research.

## References

1. M. Williamson, "Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code," HP Laboratories Bristol, Tech. Rep. HPL2002 -172, 2002.
2. H. Wang, D. Zhang, and K. G. Shin. Detecting syn flooding attacks. In Proceedings of IEEE INFOCOM '2002.
3. Seung-won Shin, Ki-young Kim, Jong-soo Jang: D-SAT: Detecting SYN Flooding Attack by Two-Stage Statistical Approach. Proceedings of the The 2005 Symposium on Applications and the Internet (SAINT'05), pp.430-436.
4. J. Mirkovic, G. Prier, and P. Reiher. "Attacking DDoS at the Source." In Proceedings of the ICNP 2002, November 2002.
5. J. Mirkovic, D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks, Ph.D. Thesis, UCLA, August 2003.

# Port Scan Behavior Diagnosis by Clustering

Lanjia Wang<sup>1</sup>, Haixin Duan<sup>2</sup>, and Xing Li<sup>1</sup>

<sup>1</sup> Department of Electronic Engineering, Tsinghua University,  
Beijing, 100084, P.R. China  
wanglanjia@ns.6test.edu.cn  
xing@cernet.edu.cn

<sup>2</sup> Network Research Center, Tsinghua University,  
Beijing, 100084, P.R. China  
dhx@cernet.edu.cn

**Abstract.** Detecting and identifying port scans is important for tracking malicious activities at early stage. The previous work mainly focuses on detecting individual scanners, while cares little about their common scan patterns that may imply important security threats against network. In this paper we propose a *scan vector model*, in which a scanner is represented by a vector that combines different scan features online, such as target ports and scan rate. A center-based clustering algorithm is then used to partition the scan vectors into groups, and provide a condense view of the major scan patterns by a succinct summary of the groups. The experiment on traffic data gathered from two subnets in our campus network shows that our method can accurately identify the major scan patterns without being biased by heavy hitters, meanwhile, possessing simplicity and low computation cost.

**Keywords:** port scan detection, network security, clustering.

## 1 Introduction

Port scan, which aims to gather information about hosts in networks, is a fundamental step in today's Internet attacks as well as worm propagation. Therefore, detecting and identifying scans is useful for tracking these malicious activities at early stage to minimize damage.

It is a challenging task to detect scans, however. Scans are broadly categorized into four well known types [13]: *vertical scans*, *horizontal scans*, *coordinated scans* and *stealth scans*. In each type, advanced scan techniques can be used to evade detection.

Another problem is that even if we just focus on the common scans (such as TCP SYN scan), it is still difficult to confirm the malice of all the sources or give a comprehensive explanation of the cause of these scans. This situation is mainly due to the activity of the worms and viruses, which result in the obfuscation of network operators when they have to deal with a flood of logs of scans.

Much work has been done on scan detection. One simple class is what Snort [10] and Bro [8] follow, detecting  $N$  connections within a time interval  $T$ . The second class of techniques is statistics-based method [6,12]. Many other approaches



are built upon the observed fact that scanners are more likely to make failed connections [3,8,9]. Another research presented in [4] considers scalable detection. In addition, some work on malicious activity detection, especially worms [1,7,11,15], is also related to scan detection.

Some of the above approaches show good performance in their special scenarios. However, there are still certain weaknesses. The first is about detection accuracy. Simple threshold based methods [4,8,9,10] often generate many false alarms and focus on heavy hitters, while methods with good performance are usually complex, sensitive to parameters or confined to certain context [3,4,6,12]. In addition, few approaches consider the second problem discussed above, namely, how to deal with a flood of scan logs.

In this paper, we pay more attention on the further analysis of detected scans. Our basic idea is that many scanners behave alike, since they scan for the same or similar causes, such as worm or virus infection. Our approach aims to provide a global view of important scans and their implication by diagnosing scan behavior.

As some previous work did, we flag the hosts that have made failed connections as *suspect scanners*. To characterize scan behavior, we build a *scan vector model* in which a suspect scanner is represented by a vector. Then we cluster the scan vectors into *scan groups*. The scanners in one group have similar scan behavior, reflecting certain *scan pattern*. The groups succinctly summarize different scan patterns that imply security threats against network.

Our method works online. It processes packets sequentially and reports major scan patterns periodically. We evaluate it on eight days traffic data gathered at the ingress of two subnets in our campus network. This evaluation shows that our method can: (i) identify and summarize major scan patterns in network without being biased by heavy hitters, (ii) effectively limit false alarms, (iii) and at the same time be easily implemented due to its simple model and low computation cost.

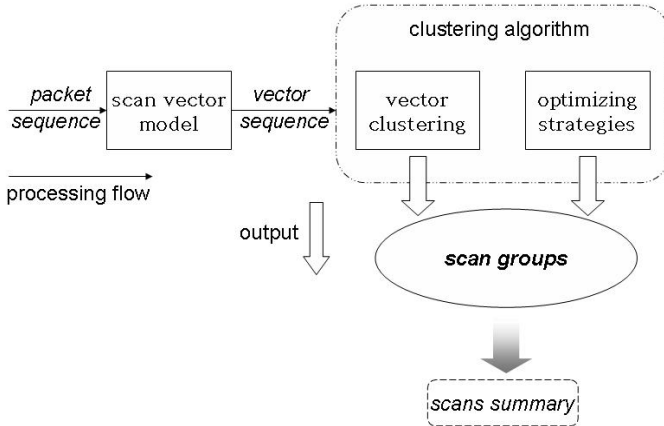
The paper is organized as follows. In Sect. 2, we show how our clustering based method diagnoses port scan behavior. Then evaluation on real traffic data is presented in Sect. 3. We discuss future work and conclude the whole paper in Sect. 4.

## 2 Diagnosis Method for Port Scan Behavior

Our method contains two parts: a *scan vector model* generating vector sequence, and an extended *center-based clustering* algorithm, which comprises primary *vector clustering* and some *optimizing strategies*. Fig. 1 shows this framework, which finally outputs a summary of major scan patterns.

### 2.1 Scan Vector Model

Method in this paper focuses on TCP SYN scans towards protected network and is based on failed connections. A *failed connection* in our method is defined as a unique destination  $\langle IP, port \rangle$  pair that a source has never successfully connected



**Fig. 1.** Framework of Scan Behavior Diagnosis Method

(“successful” means that the SYN-ACK packet is observed within a reasonable period  $T_a$ ) in a *measurement interval* (a preset time period  $I_m$ ).

We model a suspect scanner’s behavior during any numbers of measurement intervals as a vector (or a point)  $X = (x_1, x_2, \dots, x_J) = (x_1, x_2, x_3, y_0, \dots, y_I)$  ( $J = I + 4$ ) in a J-dimensional space. Each vector element  $x_j$  is the value of a feature of scan behavior as follows:

- $x_1$  *Address-related feature.* It represents the distance from the scanner’s address to the protected network address. If they are in the same /16,  $x_1 = 2$ ;  $x_1 = 1$  for the same /8; otherwise  $x_1 = 0$ .
- $x_2$  *Rate-related feature.* It is the average number of failed connections the scanner initiates in a measurement interval. So,  $x_2 > 0$ .
- $x_3$  *Targets-related feature.* It is the average number of the scanner’s destination hosts in a measurement interval. Also,  $x_3 > 0$ .
- $y_i$  *Ports-related feature.* It indicates whether port  $i$  ( $i$  is a real port number, therefore  $I = 65535$  generally) has been scanned by the scanner. If so,  $y_i = 1$ , otherwise  $y_i = 0$ .

Once a source’s first failed connection is observed, its first measurement interval starts and its vector  $X$  will be updated over time. At the end of the  $m$ th ( $m = 1, 2, \dots$ ) interval, we particularly denote  $X = X^m = (x_1^m, x_2^m, \dots, x_J^m)$ , which can be derived from its last vector  $X^{m-1}$  and increment vector  $\Delta X^m = (\Delta x_1^m, \Delta x_2^m, \dots, \Delta x_J^m)$  observed in the  $m$ th interval. According to above definitions,  $\Delta X^m$  gives the source’s target ports, the number of failed connections and destinations during its  $m$ th interval. Obviously,  $X^1 = \Delta X^1$ , and we compute  $X^m$  ( $m > 1$ ) from  $X^{m-1}$  and  $\Delta X^m$  as:

$$\begin{cases} x_1^m = x_1^{m-1} \\ x_2^m = \alpha x_2^{m-1} + (1 - \alpha)\Delta x_2^m \\ x_3^m = \alpha x_3^{m-1} + (1 - \alpha)\Delta x_3^m \\ x_j^m = \max(x_j^{m-1}, \Delta x_j^m) \quad j = 4, 5, \dots, J \end{cases} \quad (1)$$

If we denote operation (1) as  $\oplus$ , we can rewrite the whole process as:

$$\begin{cases} X^1 = \Delta X^1 \\ X^m = X^{m-1} \oplus \Delta X^m \quad m > 1 \end{cases}$$

Note that in (1),  $x_2^m$  and  $x_3^m$  are not precise average of history values. In our experiment we found that the result is not sensitive to  $\alpha \in (0, 1)$  and approximate average values can work well, so we simply set  $\alpha = 1/2$ .

In this model, we choose feature set from general knowledge and experience about scans. Vector  $X$  represents the scanner’s scan scheme, strength, desired information, etc. Method built upon current feature set is effective for scan behavior diagnosis in our experiments, while we will try to find whether better choices exist in our future work. We believe our scan vector model is a general framework and can be adopted in many scenarios by choosing appropriate features.

### 2.2 Basic Concepts Definition

Following two basic concepts are defined to be used in our clustering algorithm.

**Group Center.** The *center* of a scan group denotes the mean value of vectors in it. Suppose group  $c$  has  $N_c$  points  $\{X_1, X_2, \dots, X_{N_c}\}$ , where  $X_n = (x_{n,1}, x_{n,2}, \dots, x_{n,J})$  ( $n = 1, 2, \dots, N_c$ ). The center of group  $c$ ,  $X_c = (x_{c,1}, x_{c,2}, \dots, x_{c,J})$ , is computed as:

$$x_{c,j} = \frac{1}{N_c} \sum_{n=1}^{N_c} x_{n,j} \quad j = 1, 2, \dots, J.$$

From this definition,  $x_{c,j}$  ( $j = 4, 5, \dots, J$ ) indicates the probability of suspect scanners in group  $c$  scanning port  $j - 4$ .

**Similarity.** Computing the *similarity*  $Sim(X_1, X_2)$  between two vectors  $X_1 = (x_{1,1}, x_{1,2}, \dots, x_{1,J})$  and  $X_2 = (x_{2,1}, x_{2,2}, \dots, x_{2,J})$  is a basic step for clustering. We define:

$$Sim(X_1, X_2) = \sum_{j=1}^J w_j sim(x_{1,j}, x_{2,j}) \quad \left( \sum_{j=1}^J w_j = 1, \quad w_j > 0 \right), \quad (2)$$

where

$$\begin{cases} \text{sim}(x_{1,1}, x_{2,1}) = 1 - \frac{|x_{1,1} - x_{2,1}|}{2} \\ \text{sim}(x_{1,j}, x_{2,j}) = \frac{\min(x_{1,j}, x_{2,j})}{\max(x_{1,j}, x_{2,j})} & (\max(x_{1,j}, x_{2,j}) > 0, \quad j = 2, 3, \dots, J) \\ \text{sim}(x_{1,j}, x_{2,j}) = 0 & (\max(x_{1,j}, x_{2,j}) = 0, \quad j = 2, 3, \dots, J) \end{cases} \quad (3)$$

The values of  $w_1$ ,  $w_2$  and  $w_3$  can be chosen arbitrarily, and  $w_j$  ( $i = 4, 5, \dots, J$ ) is calculated according to the formula:

$$w_j = (1 - w_1 - w_2 - w_3) \frac{\max(x_{1,j}, x_{2,j})}{\sum_{k=4}^J \max(x_{1,k}, x_{2,k})} \quad j = 4, 5, \dots, J \quad (4)$$

This definition means that the similarity between two vectors is the weighed sum of similarities between vector element pairs. Combining (2), (3) and (4) yields:

$$\begin{aligned} \text{Sim}(X_1, X_2) &= \sum_{j=1}^3 w_j \text{sim}(x_{1,j}, x_{2,j}) + \frac{w_0 \sum_{j=4}^J \min(x_{1,j}, x_{2,j})}{\sum_{j=4}^J \max(x_{1,j}, x_{2,j})} \\ &(\sum_{j=0}^3 w_j = 1, \quad w_j > 0) \end{aligned} \quad (5)$$

For arbitrary points  $X_1$  and  $X_2$ , we have the similarity  $\text{Sim}(X_1, X_2) \in (0, 1]$ . It quantifies the resemblance between two types of scan behavior. The closer the similarity is to 0, the less similar they are.  $\text{Sim}(X_1, X_2) = 1$  is equivalent to  $X_1 = X_2$ , indicating same behavior. The larger similarity implies larger probability of the same scan intent or cause, e.g. two scanners aiming to the same service, using the same tool or infected by the same worm.

### 2.3 Vector Clustering

Primary vector clustering is the principle component of our center-based clustering algorithm. Suppose there are  $L$  scan groups,  $X_l$  denotes the center of group  $l$  and group  $l$  contains  $N_l$  vectors. For any suspect scanner  $X$ , once  $X^m$  is generated, we cluster it as one of the following two cases.

(i)  $m = 1$

That means no point representing this scanner already exists. We compute the similarity between  $X^m$  and the center of each group  $l$  ( $l = 1, 2, \dots, L$ ), then pick out group  $v$  whose center  $X_v$  is the most similar to  $X^m$  as:

$$\text{Sim}(X^m, X_v) = \max_{1 \leq l \leq L} (\text{Sim}(X^m, X_l)) .$$

Suppose  $T_s$  is a preset threshold, then:

- If  $\text{Sim}(X^m, X_v) > T_s$ ,  $X^m$  will be put into group  $v$  and  $X_v$  should be adjusted to  $X_{v'} = (x_{v',1}, x_{v',2}, \dots, x_{v',j})$  as:

$$x_{v',j} = \frac{N_v x_{v,j} + x_j^m}{N_v + 1} \quad (j = 1, 2, \dots, J) .$$

Meanwhile, updated group  $v$  has  $N_{v'} = N_v + 1$  points.

- If  $Sim(X^m, X_v) \leq T_s$ , a new group  $u$  will be created, with its center  $X_u = X^m$  and one member  $X^m$ .

(ii)  $m > 1$

Here  $X^{m-1}$  already belongs to certain group  $v$  with center  $X_v$ . What we need to do is replacing point  $X^{m-1}$  by  $X^m$  and adjusting  $X_v$  as:

$$x_{v',j} = x_{v,j} + \frac{x_j^m - x_j^{m-1}}{N_v} \quad (j = 1, 2, \dots, J).$$

In this case, a question may arise that as any scanner's point is moving all along, should we reconsider which group it belongs to? Our scan vector model and clustering algorithm guarantee the distinct characters of different scan patterns, thus a point hardly has the chance to move from one group into another.

### 2.4 Optimizing Strategies

Besides primary vector clustering, some optimizing strategies are very important in reducing noise and improving clustering performance.

**Vector Pre-checking.** This strategy works on  $\Delta X^m$  that satisfies  $\Delta x_2^m = 1$  (only one destination  $\langle IP, port \rangle$  pair) before  $X^m$  is computed. We record this  $\langle IP, port \rangle$  pair. If the pair has been recorded for more than  $T_r$  times, we add it to a list called *service set*, and if it is already in service set,  $\Delta X^m$  is ignored and  $X$  is not updated to  $X^m$ .

Simply speaking, the  $\langle IP, port \rangle$  pairs in service set are connected by a number of sources that hardly make failed connection to other destinations. Therefore, these pairs are probably services opened to public and this strategy is used to reduce false alarms.

**Group Merging.** It is possible that the first several scanners of certain scan pattern are clustered into multiple groups, for a few points cannot offer enough information. As scanners increase and more data are gathered, we can merge these groups to reduce the amount of groups and improve the accuracy of scan pattern identification.

For any group  $u$ , group merging is operated every *checking interval*  $I_c$ . We pick out group  $u$ 's most similar group  $v$  as:

$$Sim(X_u, X_v) = \max_{1 \leq l \leq L, l \neq u} (Sim(X_u, X_l)).$$

If  $Sim(X_u, X_v) > T_s$ , group  $v$  will merge group  $u$ , which means all points in group  $u$  will belong to group  $v$ , and the center will move to  $X_{v'}$  as:

$$x_{v',j} = \frac{N_v x_{v,j} + N_u x_{u,j}}{N_v + N_u} \quad (j = 1, 2, \dots, J).$$

**Group Obsoleting.** Many scan groups, especially those representing individual attackers, are *active* (being updated) only in short period. The existence of these groups increases the computation cost. So we set an *obsolete period*  $T_o$ . If a group keeps inactive longer than  $T_o$ , it will be deleted from the group set.

**Port Cutting.** Although  $X$  is a much long vector, in the implementation of our model we can only record port  $i$  that has  $y_i > 0$  by using a link, which greatly reduces the computation cost. In group center  $X_c = (x_{c,1}, x_{c,2}, x_{c,3}, y_{c,0}, \dots, y_{c,I})$ , if  $y_i$  is fairly small, port  $i$  cannot represent the essential feature of this group but increase computation cost. Therefore, we set a threshold  $T_p$ . If  $y_{c,i} < T_p$ , then set  $y_{c,i} = 0$ . This checking is operated every *checking interval*.

## 2.5 Scan Patterns Summary

Above algorithm clusters all the vectors into groups. By two types of features, these groups provide a summary of scan patterns.

The first type of features are represented by group center. Because scanners in one group have similar scan behavior, group center reveals the common character of them and we can identify a scan pattern from it.

The other type of features are statistics that assess a scan group's severity or threat on network. In our implementation, these statistics are computed along with clustering and reported together with group center every *report interval*  $I_r$ . Following six features are defined, of which some are long term features, and others are restricted in one report interval:

- start* time when the group is created, indicates its duration.
- srcs* number of suspect scanners active (scanning) in the current report interval, reveals the prevalence of this scan pattern.
- cnnts* total number of failed connections in the current report interval, reveals the strength of this scan pattern.
- sinc* difference between the number of active sources for current and previous report interval, reveals the prevalence trends.
- tsrcs* total number of scanners since this group was created, reveals long term prevalence.
- tcnts* total number of failed connections since this group was created, reveals long term strength.

As mentioned above, normal network activities also generate failed connections. Because such connections are mostly random and independent, they are much likely to be clustered into small groups with a few scanners and connections. Since the feature *tcnts* reveals the strength of a scan pattern, we can simply use it to select *major groups* and corresponding *major scan patterns*. If the value of a group's feature *tcnts* is larger than threshold  $T_m$ , it is a major group.

In summary, a combination of the two types of features describes the behavioral characteristics and assesses the severity or threat of each scan pattern. Summary of all major groups outlines a scene of port scans in networks, implying certain aspects of network security situation or trends.

### 3 Evaluation

The evaluation in this section will validate the low false alarm rate, large detection coverage and ability of scan pattern identification of our method. In addition, some issues related to implementation will also be discussed.

#### 3.1 Data Description

We use three datasets gathered at the ingress of two subnets (A and B) in Tsinghua University campus network, both have an address space of  $3 * /24$ , with average daytime bandwidth of 100Mbps. Each trace  $l$ , a line in a dataset, representing an inbound SYN packet or an outbound SYN-ACK packet, contains the fields of time stamp  $t$ , source IP  $s$ , source port  $q$ , destination IP  $d$ , destination port  $p$  and TCP flag  $f$ , thus each trace can be written as a 6-tuple  $l = \langle t, s, q, d, p, f \rangle$ . Both of the two subnets have only one ingress, so we can observe bidirectional packets of a connection. Table 1 summarizes our datasets.

**Table 1.** Summary of datasets

Dataset	Period	SYN Packets	SYN-ACK Packets
A-1	Nov 23-Nov 24	4,146,139	356,754
A-2	Mar 4-Mar 7	5,290,970	492,330
B-1	Nov 23-Nov 24	4,507,755	346,431

In our evaluation procedure, each trace is processed sequentially. Values of all the parameters are summarized in Table 2 .

**Table 2.** Summary of parameter values

Parameter	Value	Parameter	Value
$I_m$	5 minutes	$T_s$	0.5
$I_c$	5 minutes	$T_r$	3
$I_r$	5 minutes	$T_p$	0.05
$w_1$	0.1	$T_m$	10
$w_2$	0.15	$T_a$	3 seconds
$w_3$	0.05	$T_o$	2 hours
$\alpha$	0.5		

#### 3.2 Result Analysis

According to our method, a summary of major active scan groups is reported periodically. Table 3 excluding group 8 is a report sample for dataset A-1. In the ‘‘Scan Behavior’’ part, ‘‘B’’, ‘‘A’’ and ‘‘R’’ in ‘‘ $x_1$ ’’ column respectively represents feature  $x_1 = 2, 1$  and others. Due to page limit, we will only present the analysis on the results of dataset A-1, and the results of other two datasets are similar.

**Table 3.** A report sample (reported at Nov 23 16:01:07)

Scan Behavior					
Group No.	$x_1$	$x_2$	$x_3$	port $i$ ( $y_i \geq 0.8$ )	port $i$ ( $0 < y_i < 0.8$ )
1	B	8.7	8.7	445	44445 135 1957
2	B	13.5	13.4	135	445
3	B	3.0	2.2	1023 5554	1022 445 44445
4	A	3.1	1.0	6129 3127	2745 80
5	R	92.0	92.0	4899	
6	R	5.7	5.7	21	248
7	R	1.5	1.5	1433	
8	<i>R</i>	<i>1.0</i>	<i>1.0</i>	<i>23672</i>	

Severity Assessment						
Group No.	<i>srcs</i>	<i>cnnts</i>	<i>sinc</i>	<i>tsrcs</i>	<i>tcnts</i>	<i>start</i> (Nov 23)
1	58	3060	-2	369	308638	00:01:44
2	5	232	-1	51	40819	07:00:58
3	3	11	0	10	375	12:06:55
4	2	4	1	100	317	07:31:47
5	1	92	1	1	92	15:57:24
6	1	3	1	19	956	10:47:05
7	1	1	0	37	58	08:24:53
8	<i>1</i>	<i>2</i>	<i>1</i>	<i>21</i>	<i>46</i>	<i>15:31:59</i>

**False Alarm Analysis.** We investigated into all the 37 major groups through out dataset A-1. Table 4 summarizes them as 4 parts and 10 subcategories. Note that some groups may actually represent one scan pattern emerging at different time, due to *group obsoleting* operation. The 31 groups in the first three parts are important scan patterns in networks, with different intents or causes.

However, the last part is undetermined. Groups with port 113 may be normal authentication service accesses, and the last four groups with non-well-known ports possibly represent normal applications such as P2P. Thus, the last 6 groups may be false alarms. However, since they only involve 1.2% of all the scanners and much less portion of the connections, the false judgements hardly influence our macro assessment on network security.

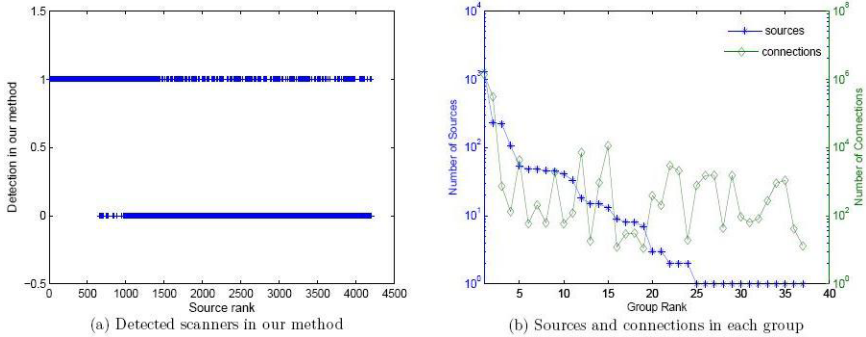
In fact, the design of *service set* greatly reduces false alarms. Group 8 in Table 3 is a false alarm generated if service set is not performed. We looked over all the reports and found some similar cases. Totally, when service set is performed, the number of sources in dataset A-1 judged as scanner is reduced by 42%, of groups is reduced by 45%. In other two datasets the results are similar.

**Detection Coverage.** We use following method to obtain a ground truth and compare our approach's result with it: (i) we set *measurement interval* to the whole period of the dataset and pick out all the suspect scanners, (ii) then we sort the scanners in descending order of the number of failed connections. Here



**Table 4.** Investigation on major active groupes of dataset A-1

	typical target ports	$x_1$	groupes	scanners	connections
<b>Vulnerability</b>	445,139	B	3	1307	1358768
	135	B	4	261	325111
	6129,3127,1433,...	A,R	9	567	1408
<b>Related</b>	1023,5554,9898	B,R	7	64	9484
	4899	R	3	3	236
<b>Service Searching</b>	21	R	2	60	2788
	80,1080,8080,...	R	1	1	273
<b>By Administrator</b>	some important ports	B	2	2	1986
<b>Undetermined</b>	113	R	2	17	43
	non-well-known ports	R	4	11	84
<b>Total</b>		—	37	2293	1700181



**Fig. 2.** Scanners detection and distribution in groups

the hypothesis is that failed connections provide strong evidence of scans, and the much long measurement interval greatly decreases the false alarms caused by failed normal service accesses.

Fig. 2(a) shows the detection coverage of our method. The  $x$ -axis is the rank of sources in the above order, and the  $y$ -axis shows whether a source is detected by our method. If the capacity of operators paying attention and taking steps on suspect scanners is 656 or less (in fact, the 656th scanner makes only 8 failed connections and 656 scanners are fairly an large amount to deal with), our method’s detection coverage is 100%. On the other hand, many scanners that make a few failed connections are detected in our method, for we make use of the correlations between scanners’ behavior other than just the number of connections. Therefore, our method can detect not only the heavy scanners but also the scanners ignored in the usual scan detection deployment. This point will be illustrated more specifically in the next subsection.

**Scan Pattern Identification.** From Table 3 we can see that 7 major active groups briefly summarize 587 scanners (71 are active at the moment), and each has its own distinct features.

Group 1 and 2 are related to common worm (Sasser, Blaster, Nachi, etc) scans. Group 3 are probably backdoor scans performed by viruses or attackers. Group 4 also represents worm (Phatbot and its variants) scans. Port 4899 of group 5 is a frequently used backdoor port, port 21 of group 6 is the most common FTP port and port 1433 of group 7 is for Microsoft SQL-Server.

Through out the whole dataset A-1, Table 4 reveals that subcategory 1 and 2 occupy more than 99% of the scan traffic due to their scan preference for addresses in the same /16, while subcategory 3 (including group 4 in Table 3) occupies more than 25% of the scanners. Although the scanners in subcategory 3 are far away from the protected subnet, they imply potential threat, therefore we think they are worthy of notice. However, since each scanner initiates only a few connections to protected subnet on average, approaches just analyzing a single scanner's behavior probably miss such scans.

Therefore, it is an important advantage that our clustering based method prevents important scan patterns – especially those with low scan rate or total connections – from being biased by certain scan patterns involving most heavy hitters. Concept of scan group provides an effective way to assess importance or threat of scans. Many small scans (a few connections observed) can compose a noticeable group, representing important scan pattern, while small (non-major) groups are really negligible.

### 3.3 Discussion

Fig. 2(b) plots the rank of each major group against its number of scanners and failed connections. Except the first two groups with both dominant scanners and connections, other groups have no proportional relation between their numbers of scanners and connections, which reveals the necessity of characterizing scan behavior with more than one feature. Also, network administrators should combine various features and their most concerns on network security to assess the importance of certain scan patterns.

We have mentioned four basic types of scans: vertical scans, horizontal scans, coordinated scans and stealth scans. The latter two are difficult to detect in most previous work. Although there are no instances in this evaluation, we believe in our method's ability in identifying a great part of such scans. Probably, the scanners participating in one attack of scan have similar behavior and are clustered into one scan group. Then adding in scanners' address feature, we may identify this coordinated scan episode. For stealth scans, they are difficult to detect because of their low scan rate. In our method, obsolete period  $T_o$  is 2 hours. Thus, as long as scan rate is larger than 0.5 failed connections per hour, the scan can be detected and attract attention when it belongs to a major group.

Computation cost is another major issue we are concerned about. The cost relies on many factors, such as bandwidth, traffic structure and total scans. Now our method has already been implemented as an online functional module in the security monitoring system for subnet A and B. In this evaluation, the computation on each dataset requires about 15 minutes on a 2.6GHz Intel-based

PC. Thus from the point of computation cost, our approach has the potential to scale up to higher speed environment.

## 4 Conclusion and Future Work

In this paper, we have proposed an approach to diagnose port scan behavior. Our work aims to provide a succinct summary of important scan patterns in networks, which is useful for effectively monitoring network security, but little explored in the pervious work. Our method is based on the fact that scans have strong correlations because of scanners' same or similar intents. We model any suspect scanner as a moving point in a high dimensional space and a center-based clustering algorithm clusters scanners of similar behavior into one scan group, which represents a major scan pattern in networks.

We evaluate our method by real network data. All important scan patterns in our datasets are identified, with negligible false alarms and low computation cost. The results validate our method's ability in effectively diagnosing port scan behavior in networks.

In the future work, we will go on researching on how to design an optimal vector that catches more essential characters of scan behavior. Beyond TCP SYN scan, other scan techniques [14] will be also taken into account. Furthermore, as the information of any scan pattern reported at intervals also forms a timeseries, we will try to find whether some forecast models that have been widely studied and applied for traffic anomaly detection [2,5] can work on this scan related timeseries and draw meaningful conclusions about its developing trends.

## Acknowledgement

This work is supported in part by the National High Technology Research and Development Program of China (863 Program) under Grant No. 2003AA142080 and the National Natural Science Foundation of China under Grant No. 60203004. The authors are grateful to Jianguang Di and Xueli Yu for their help in building experiment environment.

## References

1. V.H. Berk, R.S. Gray, and G. Bakos: Using sensor networks and data fusion for early detection of active worms. In Proceedings of the SPIE AeroSense, 2003
2. J. Brutlag: Aberrant Behavior Detection in Timeseries for Network Monitoring. In Proceedings of USENIX Fourteenth Systems Administration Conference (LISA), New Orleans, LA, Dec 2000
3. J. Jung, V. Paxson, A. W. Berger, H. Balakrishnan: Fast Portscan Detection Using Sequential Hypothesis Testing. In Proceedings of 2004 IEEE Symposium on Security and Privacy, pages 211–225, Berkeley, CA, USA, May 2004
4. R. R. Kompella, S. Singh, and G. Varghese: On Scalable Attack Detection in the Network. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pages 187–200, Taormina, Sicily, Italy, Oct 2004

5. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen: Sketch-based Change Detection: Methods, Evaluation, and Applications. In Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, Pages: 234–247, Miami Beach, FL, USA, Oct 2003
6. C. Leckie and R. Kotagiri: A probabilistic approach to detecting network scans. In Proceedings of the Eighth IEEE Network Operations and Management Symposium (NOMS 2002), pages 359–372, Florence, Italy, Apr 2002
7. D. Moore, C. Shannon, G. M. Voelker, and S. Savage: Internet Quarantine: Requirements for Containing Self-Propagating Code. In Proceedings of IEEE INFOCOM, Apr 2003
8. V. Paxson: Bro: A System for Detecting Network Intruders in Real Time. In Proceedings of the 7th USENIX Security Symposium, 1998
9. S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo: Surveillance detection in high bandwidth environments. In Proceedings of the 2003 DARPA DISCEX III Conference, pages 130–139, Washington, DC, Apr 2003
10. M. Roesch: Snort: Lightweight intrusion detection for networks. In Proceedings of the 13th Conference on Systems Administration (LISA-99), pages 229–238, Berkeley, CA, Nov 1999. USENIX Association
11. S. E. Schechter, J. Jung, A. W. Berger: Fast Detection of Scan Worm Infections. In Proceedings of the Seventh International Symposium on Recent Advances in Intrusion Detection, Sophia Antipolis, France, Sep 2004
12. S. Staniford, J. A. Hoagland, and J. M. McAlerney: Practical automated detection of stealthy portscans. In Proceedings of the 7th ACM Conference on Computer and Communications Security, Athens, Greece, 2000
13. V. Yegneswaran, P. Barford, and J. Ullrich: Internet intrusions: global characteristics and prevalence. In Proceedings of the 2003 ACM SIGMETRICS, volume 31, 1 of Performance Evaluation Review, pages 138–147, New York, Jun 2003. ACM Press
14. M. de Vivo, E. Carrasco, G. Isern and G. de Vivo: A Review of Port Scan Techniques. Computer Communications Review, 29(2), April 1999, pages 41–48
15. C. C. Zou, L. Gao, W. Gong, and D. Towsley: Monitoring and Early Warning for Internet Worms. In Proceedings of the 10th ACM conference on Computer and communications security, Washington, DC, USA, Oct 2003

# Network Vulnerability Analysis Through Vulnerability Take-Grant Model (VTG)

Hamid Reza Shahriari, Reza Sadoddin, Rasool Jalili,  
Reza Zakeri, and Ali Reza Omidian

Network Security Center, Department of Computer Engineering,  
Sharif University of Technology, Tehran, Iran\*  
{shahriari, rzakery, omidian}@mehr.sharif.edu  
saededdi@ce.sharif.edu, jalili@sharif.edu

**Abstract.** Modeling and analysis of information system vulnerabilities helps us to predict possible attacks to networks using the network configuration and vulnerabilities information. As a fact, exploiting most of vulnerabilities result in access rights alteration. In this paper, we propose a new vulnerability analysis method based on the Take-Grant protection model. We extend the initial Take-Grant model to address the notion of vulnerabilities and introduce the vulnerabilities rewriting rules to specify how the protection state of the system can be changed by exploiting vulnerabilities. Our analysis is based on a bounded polynomial algorithm, which generates the closure of the Take-Grant graph regarding vulnerabilities. The closure helps to verify whether any subject can obtain an access right over an object. The application of our results have been examined in a case study which reveals how an attacker can gain an unauthorized access right by exploiting chain of vulnerabilities.

## 1 Introduction

The distribution and complexity of computer networks and the large number of services provided by them, makes computer networks vulnerable to cyber attacks. Currently several tools exist which analyze a host vulnerabilities in isolation, but to protect networks against attacks, we need to consider the overall network vulnerabilities and the dependency between services provided by the hosts.

Services may provide an acceptable level of security when considered in isolation, but a combination of these secure services may lead to subtle attack scenarios. For example, the file transfer protocol (ftp) and the hypertext transfer protocol (http) offered simultaneously in a same host, may allow the attacker to write in a web directory using the ftp service. This causes the web server to execute a program written by the attacker. Consequently, comprehensive analysis of network vulnerabilities needs considering individual hosts as well as their relationships.

The complexity of analyzing network vulnerabilities can be augmented as the number of hosts and services increases. Facing current enormous networks, automated approaches are necessary to analyze vulnerabilities.

---

\* This research was in part supported by a grant from I.P.M (No. CS1383-4-04).

Some approaches have been proposed in the literature to analyze network vulnerabilities from the point of view of the relations between individual hosts and network configurations [1], [2], [3], [4], [5]. Such approaches mainly use model checking and graph-based techniques to generate and analyze an attack graph; the task has been done in exponential time. In [6], [7] polynomial time approaches have been suggested for the same problem without any specific upper bound on polynomial degree.

In this paper, we extend the Take-Grant protection model to address the concept of vulnerabilities, which allow an entity to change the protection state of the system and violate security policies. We propose a framework to model vulnerabilities based on their preconditions and postconditions, and an algorithm to analyze the model in bounded polynomial time with the size of protection system graph. The proposed algorithm can generate possible attack scenarios as well.

The remainder of this paper is organized as follows: Firstly, the previous works on Take-Grant protection model and network vulnerability analysis are reviewed. Then, our Vulnerability Take-Grant model is introduced as an extension to the Take-Grant model. The way to exploit some vulnerabilities can be represented in the extended model is shown in section 5. Our approach to vulnerability analysis comes in the next section. The application of Vulnerability Take-Grant model in a real network will be also examined in section 7. Finally, we conclude and propose future areas of research.

## 2 Related Work

The Take-Grant protection model was first developed by Jones et al. [8] in which the *safety problem*<sup>1</sup> could be solved in linear time. They provided the necessary and sufficient conditions under which rights and information could be transferred between two entities of the protection system and a linear time algorithm to test those conditions. Applications of the Take-Grant model to various systems have been explored separately [9], [10], [11], [12], and [13]. Extending the initial Take-Grant model also has been experienced by Frank and Bishop [14]. They proposed a method of extending the Take-Grant model to add notion of the cost of information or right flows and finding the most likely path in order of costs. Besides decidability, time complexity of the deciding algorithm has also been emphasized in nearly all previous works. These features have made the Take-Grant model more attractive than other formal access control models.

Based on the authors' knowledge, the Take-Grant protection model has not been used for host or network vulnerability analysis so far. Previous approaches for network vulnerability analysis mainly used model checking and graph-based techniques whose time complexity is either exponential or polynomial. Such approaches mainly depend on some off-the-shelf tools for scanning individual host vulnerabilities. Vulnerability scanner tools such as Nessus [15] scan hosts to discover vulnerabilities in the configuration. However, they do not investigate how a combination of configurations on the same host or among hosts on the same network can contribute to the vulnerabilities.

---

<sup>1</sup> The safety problem is defined in [22] as follows: Given an initial configuration of a protection system, whether a subject  $s$  can obtain some access right  $r$  over an object  $o$ ?

The NetKuang system tries to assess beyond host vulnerabilities. It is an extension to a previous work on building a rule-based expert system, named Kuang [1]. Dacier [2] proposed the concept of privilege graphs. Privilege graphs are explored to construct an attack state graph, which represents different ways in which an intruder may reach a certain goal, such as root access on a host.

Ritchey and Ammann [3] used model checking for vulnerability analysis of networks via the model checker SMV. They could obtain only one attack corresponding to an unsafe state. The experiment was restricted to only specific vulnerabilities. However, the model checking approach has been used in some other researches to analyze network vulnerabilities [6], [16]. The model checking has the scalability problem which some researchers tried to overcome [6]. Ramakrishnan and Sekar [4] used a model checker to analyze a single host system with respect to combinations of unknown vulnerabilities. The key issue in their research was checking of infinite space model using model abstraction. Swiler et al. presented a method in [17] for generating attack graphs. Their tool constructs the attack graph by forward exploration.

In [5] CSP was used to model and analyze TCP protocol vulnerabilities. In this approach, the model checker FDR2 was used to verify some simple security properties and find attack scenarios. CSP has been used widely in modeling and analyzing security protocols [18] and verifying intrusion detection systems [19]. Noel et al. presented TVA in [7] and [20] and investigated it more in [21]. In this approach, exploits are modeled as pre/post-conditions and a specific tool has been used to construct the attack graph. Encoding each exploit individually resulted in a large and complex model.

In our approach, similar vulnerabilities are represented in a single model. For example, all buffer overflow vulnerabilities are treated similarly. Moreover, this reduces the size of the model and cost of analysis. Moreover, our approach finds the attack paths using an algorithm in bounded polynomial time with the size of protection system graph.

### 3 Take-Grant Protection Model

The Take-Grant protection model is a formal access control model, which represents transformation of rights and information between entities inside a protection system. This model was presented first by Jones et al. [8] to solve the “Safety Problem”. They showed that using Take-Grant model, the safety problem is decidable and also can be solved in linear time according to the number of subjects and objects of the system.

In this model the protection state is represented as a directed finite graph. In the graph, vertices are entities of the system and edges are labeled. Each label indicates the rights that the source vertex of the corresponding edge has over the destination vertex. Entities could be subjects (represented by ●), objects (represented by ○) or play the both roles (represented by ⊗). The set of basic access rights is denoted as  $R=\{t,g,r,w\}$  which  $t$ ,  $g$ ,  $r$  and  $w$  respectively stand for take, grant, read, and write access rights. To model the rights transfer, Take-Grant protection model uses a set of rules called de-jure rules. These rules transfer the Take-Grant graph to a new state which reflects the modification of protection state in an actual system. The de-jure

rules are *take*, *grant*, *create* and *remove*. The take and grant rules are described briefly as:

1. Take rule: Let  $x$ ,  $y$ , and  $z$  be three distinct vertices in a protection graph  $G_0$  and let  $x$  be a subject. Let there is an edge from  $x$  to  $y$  labeled  $\gamma$  where  $t \in \gamma$ , an edge from  $y$  to  $z$  labeled  $\beta$ . Then the take rule defines a new graph  $G_1$  by adding an edge to the protection graph from  $x$  to  $z$  labeled  $\alpha$ , where  $\alpha \sqsubseteq \beta$ . Fig 1.(a) shows the take rule graphically.
2. Grant rule: Let  $x$ ,  $y$ , and  $z$  be three distinct vertices in a protection graph  $G_0$  and let  $x$  be a subject. Let there is an edge from  $x$  to  $y$  labeled  $\beta$  where  $g \in \beta$ , an edge from  $x$  to  $z$  labeled  $\beta$ . Then the grant rule defines a new graph  $G_1$  by adding an edge to the protection graph from  $y$  to  $z$  labeled  $\alpha$ , where  $\alpha \sqsubseteq \beta$ . Fig.1(b) shows the grant rule graphically.

Having the take right over another subject or object means that its owner can achieve all rights of the associated subject or object unconditionally. However, obtaining the rights through the grant rule requires cooperation of the grantor.

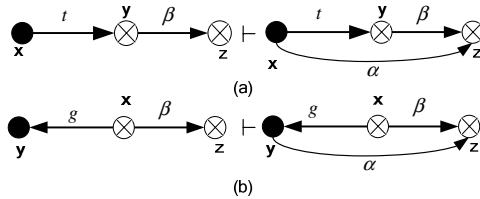


Fig. 1. (a) take rewriting rule. (b) grant rewriting rule.

### 4 The Vulnerability Take-Grant Model

The initial Take-Grant model is extended to address the notion of vulnerability. To use advantages of the Take-Grant model, it is critical to preserve the model abstraction. Without loss of generality, just for simplicity, here we only consider vulnerabilities which increase the attacker access rights.

The set of all possible vulnerabilities for a single host (which henceforth will be referred as *VLN*) can be found easily using vulnerability scanner tools such as Nessus. The *vulnerability* function associates a set of vulnerabilities to each vertex. More formally:

$$vulnerability: V \rightarrow 2^{VLN} \tag{1}$$

where  $V$  stands for the Vulnerability Take-Grant graph vertices and  $2^{VLN}$  is the power set of *VLN*.

Henceforth, we refer to Vulnerability Take-Grant graph as VTG graph. Beside the initial Take-Grant rights, we need the following access rights:



1. **x**, which represents the execution right of a subject over an object.
2. **o**, which stands for ownership and represents the ownership of a subject over an object. This right specifies which subject currently owns an object.
3. **h**, which stands for hosting and represents a machine hosts an entity.

Thus, we extend the right set to be  $R = \{t, g, r, w, x, o, h\}$ .

We define the function *rights* to show the set of rights each entity has over another entity. More formally:

$$rights(u,v): V \times V \rightarrow 2^R \quad (2)$$

In this model, vulnerabilities of each entity are denoted by the label of related vertex. We present some examples of the model in the next section.

## 5 Modeling Vulnerabilities

The Vulnerability Take-Grant model is used to model vulnerabilities which their exploit can be demonstrated by a change in access rights. The change is represented by some rules we call them *vulnerability rewriting rules (VRR)*. To demonstrate how vulnerabilities can be modeled using VTG, some groups of vulnerabilities are used as examples following by their graphical representation. In later sections of this paper, we focus more on the model.

### 5.1 Buffer Overflow Vulnerabilities

Buffer overflow vulnerabilities (BOF) are reported as the most exploited ones among network attack [23]. We model all vulnerabilities of this type as a rewriting rule. Assume a process  $p$  (having BOF) is running on the host  $m$  with the privilege of user account  $a$ ; and the attacker  $A$  has the execution right over  $p$ . Now  $A$  can exploit BOF and execute his arbitrary code with the privilege of the user account  $a$ .

Fig. 2(a) depicts the *buffer overflow rewriting rule* and demonstrates how exploiting the *BOF* vulnerability results in a change in access rights. As shown, after exploiting *BOF*, the attacker achieves the new *take* access right ( $t$ ) over user account  $a$ . We use the notation  $\{BOF\}$  as a vertex label to represent this vulnerability.

### 5.2 Weak Password Vulnerability

The weak password vulnerability (*WP*) arises when a user account with a weak password exists on a host  $m$  and the host provides a *login* service to other users (similar to what is common in web-based services). Assume the user  $u$  has an account  $a$  on host  $m$  and has chosen a weak password for it. Also assume this host provides a login service which provided by process  $p$ . Now the attacker  $A$  can guess the password of user  $u$  and take all the privileges of user account  $a$ .

Fig. 2(b) depicts the *password cracking rewriting rule* and demonstrates how exploiting the *WP* vulnerability results in a change in access rights. As shown, after exploiting *WP*, the attacker achieves the *take* access right ( $t$ ) over user account  $a$ . We use the notation  $\{WP\}$  as a vertex label to represent this vulnerability. In addition, we

use the vertex label  $\{Login\}$  to show the login service provided by process  $p$ . In fact, providing the login service is not a vulnerability, but the same notation is used for Fig. 2(b) depicts the *password cracking rewriting rule* and demonstrates how exploiting the *WP* vulnerability results in a change in access rights. As shown, after exploiting *WP*, the attacker achieves the *take* access right ( $t$ ) over user account  $a$ . We use the notation  $\{WP\}$  as a vertex label to represent this vulnerability. In addition, we use the vertex label  $\{Login\}$  to show the login service provided by process  $p$ . In fact, providing the login service is not a vulnerability, but the same notation is used for vulnerabilities and services to preserve consistency and simplicity of the model.

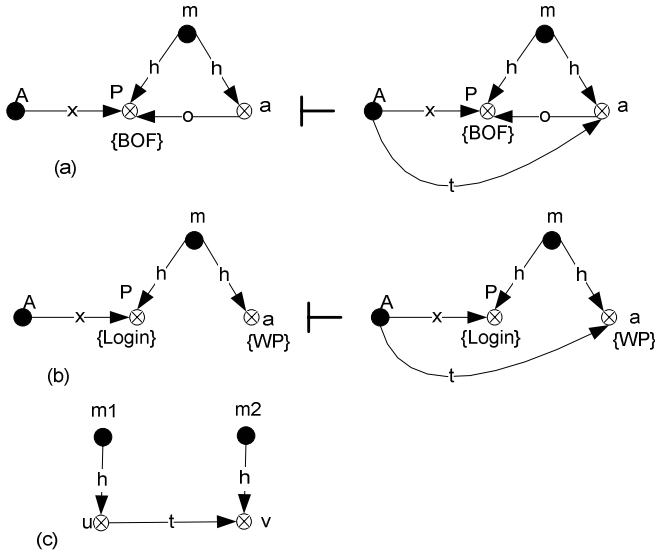


Fig. 2. Modeling vulnerabilities: (a) Buffer Overflow (b) Password Cracking (c) rhost vulnerability

### 5.3 Trust Vulnerabilities

Sometimes a user trusts another user and allows him/her to access resources. One of the best examples of such vulnerabilities is the *rhost* facility in UNIX. The *rhost* vulnerability occurs when a user trusts another user on a host or on the network. On operating systems such as UNIX and Windows NT based operating systems, users are allowed to define a list of their trustees in a file. In UNIX-based operating systems, typically this file is named *.rhosts* and is located in the user’s home directory. These trustees take all the access rights of the user who trusts them.

The attacker does not need to run any program or malicious code to exploit this vulnerability. Fig. 2(c) demonstrates how this vulnerability can be modeled in *VTG*. Assume user account  $v$  is trusted by user account  $u$ . This trust is shown in *VTG* graph by a *take* edge from  $u$  to  $v$ . The vertex label  $\{rhost\}$  is used to represent this vulnerability. It should be mentioned that this vulnerability does not need any new rewriting rule, because no action is required to exploit it and we can add the related edges and vertex labels while we are building the *VTG* graph.

## 6 Analyzing the Model

In this section, we present a method for network vulnerability analysis using VTG model and investigate its efficiency for a set of vulnerabilities. Our analysis is based on the following question:

“Is it possible for attacker **A** to achieve access right **r** over **y** or not?”

or more formally, having the initial VTG  $G_0$ , is there a VTG graph  $G_k$  having an edge in  $G_k$  labeled  $r$ , and the sequence of transitions  $\vdash^*$ , such that  $G_0 \vdash^* G_k$  ?

Rights in the *Take-Grant* protection model (and of course in VTG), can be transferred either conditionally or unconditionally. It is also the case in application of this model in vulnerability analysis. The attacker can exploit some vulnerabilities unconditionally while some others involve cooperation of other system subjects which grant some rights either unknowingly or intentionally. Our focus, here, is to consider unconditional capability of an attacker to acquire rights. To be precise, we are interested in the following question:

“Can attacker **A** achieve access right **r** over **y** unconditionally?”

Conditional transformation of rights has been investigated in the previous works on *Take-Grant* protection model. Authors in [8] and [24] dealt with this question provided that all the subjects in the system would cooperate. Snyder introduced the concept of “stealing” of rights and provided the necessary and sufficient conditions under which rights could be stolen if no owner of right  $r$  would grant it to other subjects or objects.

*Grant* rules are useless when our focus is on unconditional transformation of rights. What we mean by unconditional transformation of rights can be defined more formally in VTG by the predicate *can•access*:

**Definition 1.** The predicate *can•access*( $\alpha$ , **x**, **y**, VTG<sub>0</sub>) is true for the right  $\alpha$ , the vertex **x** (as subject), the vertex **y** (as subject or object), and the graph VTG<sub>0</sub>; if there exist protection graphs VTG<sub>1</sub>, ..., VTG<sub>n</sub> such that VTG<sub>0</sub>  $\vdash^*$  VTG<sub>n</sub> using only *take* and *vulnerability* rewriting rules, and there is an edge from **x** to **y** labeled  $\alpha$  in VTG<sub>n</sub>.

To answer the predicate *can•access*( $\alpha$ , **x**, **y**, VTG<sub>0</sub>), it is needed to construct VTG’s closure regarding to *de-jure* and *vulnerability rewriting rules*. First, we define the concept of *closure*:

**Definition 2.** Let A be the set of some rewriting rules. We define  $G^A$  the closure of G if all possible rules of A have been applied in  $G^A$  and no more rewriting rules can be applied in it.

The initial state of VTG graph is changed by both *de-jure* and *vulnerability* rewriting rules. Let’s  $G^{dejure}$  be the closure of G regarding to *de-jure* rewriting rules and  $G^{VRR}$  be the closure of G regarding to *vulnerability rewriting rules*. It may be possible to apply one set of rewriting rules after constructing a closure using the other set of rewriting rules.

To capture all the possible attack paths, a *complete* closure is needed. We use the following psudo-code to construct a *complete* closure in which all the possible rewriting rules have been applied and no new rule can be applied anymore.

### Gen\_complete\_Closure(G)

- 1- Let list  $F$  initially contain all ordered pairs of the form  $(e, r)$  where  $e$  denotes edges labeled  $t$ , and  $r$  denotes the associated right.
- 2- **While** ( $\text{! IsEmpty}(F)$ )  
//applying all possible *de-jure* rules
- 3- **While** ( $\text{! IsEmpty}(F)$ )
- 4-     Let  $(e, r) = \text{head}(F)$
- 5-     For each *take* rule applicable through  $e$
- 6-         **Add the resulting edge and its associated right to  $F$ , if it has not been inserted yet.**
- 7-     Delete  $(e, r)$  from  $F$   
//applying *BOF* rewriting rules
- 8-     **for** all  $v \in V$
- 9-         **if**  $\text{BoF} \in \text{vulnerability}(v)$  **then**
- 10-             **Add an edge labeled  $t$  from all accounts having access to  $v$  to the owner of  $v$ .**
- 11-             **Add the above edge and its associated right to  $F$ , if it has not been added yet.**  
//applying *password cracking* rewriting rules
- 12-         **for** all  $M \in \text{Hosts}$  //Hosts is the set of all machines in the system
- 13-             **Add an edge labeled  $t$  from all accounts having login access to  $M$  to accounts having weak passwords in  $M$ .**
- 14-             **Add the above edge and its associated right to  $F$ , if it has not been added yet.**

Theorem 1 deals with the correctness and time complexity of *Gen\_complete\_Closure* algorithm.

**Theorem 1.** *Gen\_Complete\_Closure* constructs the complete closure of  $G$  correctly in  $O(V^4)$ .

**Proof:** At first, we prove that lines 2-7 deal with constructing  $G_i^{\text{dejure}}$  given the input graph  $G_i$  at the beginning of the  $i$ th round of the algorithm. We should prove that the algorithm adds all the possible edges and rights and no multiple edges exist between vertices. Let  $L = \{(R_1, r_1), (R_2, r_2), \dots, (R_n, r_n)\}$  be a sequence of applied rules leading to a correct  $G_i^{\text{dejure}}$  closure, where  $R$  and  $r$  stand for related *rules* and *rights* respectively. Assume there are some rights in  $L$  which are not produced by our algorithm and let  $(R_k, r_k)$ ,  $1 \leq k \leq n$ , to be the first such ordered pair appearing in  $L$ . We define the rights  $t$  in Fig. 1 the *basic right* of the *take* rule. The *basic right* of  $R_k$  should have been already added to graph by one of the rules  $R_1$  to  $R_{k-1}$ . These rules have been applied by our algorithm similarly; so the *basic right* of  $R_k$  has been added to  $F$  and should be considered by the algorithm which leads in addition of  $r_k$  and contradicts the initial assumption that  $r_k$  has not been added by *Gen\_complete\_Closure* Algorithm. Moreover, the condition of line 6 in the algorithm makes sure that no ordered pair will be added to  $F$  repeatedly.

No we show that lines 9-14 of the algorithm constructs  $G_i^{VRR}$  (closure of  $G$  regarding to *buffer overflow* and *password cracking* vulnerability rewriting rules) correctly given the input graph  $G_i$  in the  $i$ th round of the algorithm. It is obvious that all the *buffer overflow* and *password cracking* rewriting rules are applied once by the algorithm. It's sufficient to prove that there is no need to consider any vulnerable vertex in  $VTG$  more than once. The applied rewriting rules add an edge labeled  $t$  to  $VTG$ . This operation doesn't make a previously considered vertex a candidate for applying a new *vulnerability* rewriting rule, because having an edge labeled  $t$  is not a part of precondition of any *vulnerability* rewriting rule.

No multiple rights (and their associated edges) will be added by algorithm, hence the list  $F$  will contain  $O(V^2)$  ordered pairs at most. To apply the necessary *take* rules in line 5, it is sufficient to consider all the adjacent edges to the current edge  $e$ , and it will take  $O(V)$  at most. The cost of adding new edges and their associated rights would be of  $O(1)$  because it only requires checking the associated edges in the constructed graph. Every edge and its associated right will be added to and removed from list  $F$  at most once, thus time complexity of lines 2-7 is  $O(V^3)$  in overall. The cost of applying *buffer overflow* and *password cracking* rewriting rules will be of  $O(V)$  and  $O(V^2)$ , respectively. We have just shown that the outer loop of the algorithm will be executed at most  $V^2$  times. Thus the time complexity of lines 8-14 will be of  $O(V^4)$ . Consequently, the time complexity of the algorithm is  $O(V^4)$ . ■

Having a *complete* closure, we can answer the *can•access* predicate which was defined at the beginning of this section in  $O(1)$ . Therefore the following theorem holds:

**Theorem 2.** Let  $A$  be the union of the *take* and *vulnerability* rewriting rules. We can construct  $G^A$  in polynomial time and verify the *can•access* predicate in constant time.

It is worthy of note that the initial cost of constructing the *complete* closure will be paid once and the attacker's capability to access the network resources can be answered in constant time afterwards. Moreover, the algorithm can be modified to generate *attack path*. The *attack path* can be tracked by assigning text labels to rights when applying rewriting rules. The assigned text describes how the vulnerabilities are exploited or the *de-jure* rules are applied as well as the subjects and objects involved in the rules. Fig. 3 depicts how we can generate a new label from two previously generated ones. Assume that rights  $p$  and  $q$  have been already added by rewriting rules and text labels  $Label(p)$  and  $Label(q)$  contain the attack scenarios which lead to addition of these rights respectively. Moreover, assume we can now apply a new rewriting rule and obtain the new right  $r$ . The associated text label of  $r$ ,  $Label(r)$ , can be of the following form:

$Label(r) = \{Label(p), Label(q), \text{"having access rights } p \text{ and } q, \text{ we can apply rewriting rule } x \text{ and achieve right } r" \}$

Subsequently,  $Label(r)$  contains the complete attack scenario acquiring right  $r$ .

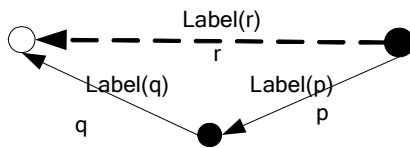


Fig. 3. Generating attack scenario labels

## 7 Case Study

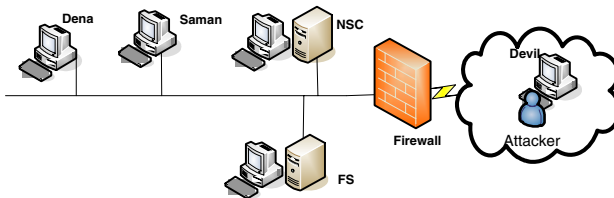
In this section, we represent the application of Vulnerability Take-Grant model and the acquired results in vulnerability analysis of a typical network. Besides the previously introduced rewriting rules, we need some general rules to analyze the real world vulnerabilities. One of these general rules which addressed here arises from the fact that each user's access rights are subset of root's access rights. This fact can be shown in VTG model as a set of *take* edges drawn from *root* account to other user accounts defined on the same host.

Fig. 4 shows a local network. The attacker is outside the network. The firewall configuration allows remote users to just have access to web and mail services. The attacker goal is to gain access to *Ali*'s files hosted on *Saman*. On the machine *NSC*, *HTTP* and *SMTP* services are listening to the associated ports. These services are running with the user privileges *apache* and *root* respectively. Also *SSH* and *SMB* services are running on the machine *FS* with user privilege *root* and *RPC* service is running on *Saman* with the same user privilege.

Using the Nessus scanner, we found that the services *HTTP* on *NSC*, *SMB* on *Saman* and *RPC* on *FS* have buffer overflow vulnerability. Moreover, we found that the user account *root* on the machine *FS* suffers from weak password vulnerability and the user *Ali* has added the account manager from machine *FS* to its *.rhost* file.

This network's VTG model is represented in Fig. 5. To avoid congestion, unnecessary relations between hosts are ignored in the figure, and the new rights added as the impacts of the vulnerabilities are showed by dotted edges, and the attacker final path is showed by dashed edges.

By using *Gen\_Complete\_Closure* algorithm described in the previous section and applying the rewriting rules on the above VTG graph,  $G^A$  is generated.



**Fig. 4.** The example network topology

As mentioned above, the Attacker's goal is to access *Ali*'s file on the *Saman*. Attacker is allowed to access *Ali*'s file if and only if there is an edge from Attacker to *Ali* in  $G^A$  including right *r* in its set of access rights. The attack path which brings the Attacker to the *Ali*'s file is shown in dashed line in Fig. 5. We can obtain the attack path by using the previously described technique. One possible attack scenario is as follows:

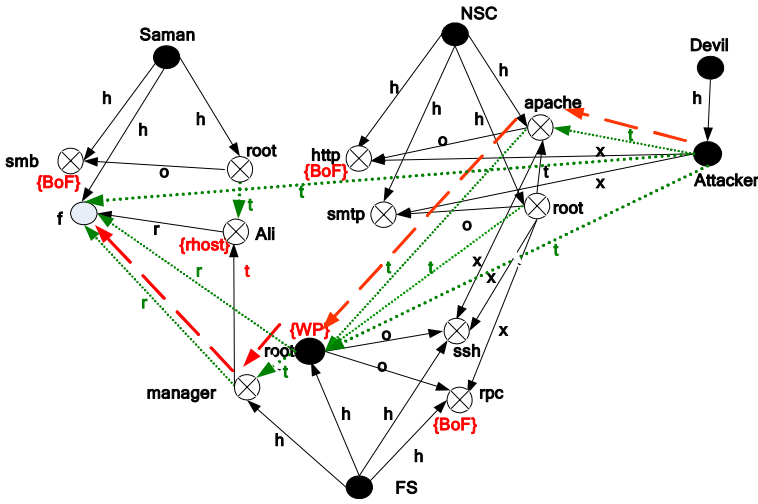


Fig. 5. Part of  $G^A$ , generated for the case study network using *Gen\_Complete\_Closure*

1. The *Attacker* exploits the *HTTP* buffer overflow vulnerability on the machine *NSC* and gains the user privilege *apache* on this machine.
2. Now the *Attacker* has access to *SSH* service on machine *FS* and can try to guess *root* password.
3. After finding the *root* password, the *Attacker* has all the rights of user account *manager* on machine *FS*.
4. Pretending to be *manager*, the *Attacker* acquires *Ali*'s access rights on machine *Saman*.
5. Consequently, the *Attacker* reaches its final goal, which is having access to file *f* on machine *Saman*.

## 8 Conclusions and Future Works

In this paper, we introduced a new method for network vulnerability analysis which is based on the Take-Grant protection model. This method affords the possibility of representing the protection state of a network with a formal model. The attacker's capability to access the resources of network can be analyzed by the model. We also introduced the *complete* closure concept to address all the possible ways of exploiting vulnerabilities and presented an algorithm to construct the *complete* closure graph in  $O(V^4)$ . With *complete* closure, the safety problem could be answered in constant time. Besides analyzing vulnerabilities, the proposed method could generate possible attack scenarios.

It is possible to use the model for more comprehensive analysis. Answering to questions such as finding the *critical vulnerable path*, finding the *shortest path of accessing a right* and finding *minimum cost path of accessing rights* (considering the

possibilities or difficulties of exploiting different vulnerabilities) can represent further applications of Take-Grant model in vulnerability analysis. Reducing the time complexity of the analysis can be considered as well. The proposed algorithm constructs the *complete* closure in bounded polynomial time and answers to *safety problem* in constant time. Considering the similarity of *de-jure* and *vulnerability* rewriting rules, it may be possible to analyze the vulnerabilities by an algorithm just like *can•steal* in linear time. The nature of Take-Grant model makes it most suitable for analyzing the vulnerabilities based on changes in access rights. Extending this model to cover a broader set of vulnerabilities will be of particular interest. This suggests several avenues of research. First, it can be studied how to model the vulnerabilities which decrease the access rights. Secondly, it is interesting to generalize this method for analyzing vulnerabilities based on a suitable taxonomy of vulnerabilities and their preconditions and postconditions.

## References

1. D. Zerkle, and K. Levitt: NetKuang – A Multi-Host Configuration Vulnerability Checker. Proceedings of the sixth USENIX UNIX Security Symposium, San Jose, CA, 1996.
2. M. Dacier, Y. Deswarte: Privilege Graph: An Extension to the Typed Access Matrix Model. Proceedings of Third European Symposium on Research in Computer Security (ESORICS 94), (Brighton, UK), Lecture Notes in Computer Science: Computer Security, 875, pp.319-334, Springer-Verlag, 1994.
3. R.W. Ritchey, P. Ammann: Using Model Checking to Analyze Network Vulnerabilities. Proceedings of IEEE Symposium on Security and Privacy, pages 156–165, May 2001.
4. C.R. Ramakrishnan, R. Sekar: Model-Based Analysis of Configuration Vulnerabilities. Journal of Computer Security, vol. 10, no. 1/2, pp. 189-209, 2002.
5. H. R. Shahriari, R. Jalili: Using CSP to Model and Analyze Transmission Control Vulnerabilities within the Broadcast Network. Proceedings of the IEEE International Networking and Communication Conference (INCC'2004), June 2004, pp. 42-47.
6. P. Ammann, D. Wijesekera, S. Kaushik: Scalable Graph-Based Network Vulnerability Analysis. Proceedings of 9th ACM Conference on Computer and Communications Security, Washington, DC, November 2002.
7. S. Noel, B. O'Berry, C. Hutchinson, S. Jajodia, L. Keuthan, A. Nguyen: Combinatorial Analysis of Network Security. Proceedings of the 16th Annual International Symposium on Aerospace/Defence Sensing, Simulation, and Controls, Orlando, Florida, April 2002.
8. J. R. Lipton, L. Snyder: A Linear Time Algorithm for Deciding Security. Proc 17<sup>th</sup> Annual Symp. on the Foundations of Computer Science (Oct. 1976), 33-41.
9. M. Bishop: Hierarchical Take-Grant Protection Systems. Proc. 8th Symp. on Operating Systems Principals (Dec. 1981), 107-123.
10. A. Jones: Protection Mechanism Models: Their Usefulness. in Foundations of Secure Computing, Academic Press, New York City, NY (1978), 237-254
11. L. Snyder: On the Synthesis and Analysis of Protection Systems. Proc. Sixth Symp. on Operating Systems Principals (Nov. 1977), 141-150.
12. M. Wu: Hierarchical Protection Systems. Proc. 1981 Symp. On Security and Privacy (Apr. 1981), 113-123.
13. M. Bishop: Conspiracy and Information Flow in the Take-Grant Protection Model. Journal of Computer Security, vol 4(4), 1996, pp 331-360.
14. J. Frank, M. Bishop: Extending The Take-Grant Protection System. Technical Report, Department of Computer Science, University of California at Davis, 1996.



15. R. Derasion, [online]: The Nessus Attack Scripting Language Reference Guide. 2000. Available from: <http://www.nessus.org>.
16. O. Sheyner, J. Haines, S. Jha, R. Lippmann, J. Wing: Automated Generation and Analysis of Attack Graphs. Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, 2002.
17. L. Swiler, C. Phillips, D. Ellis, S. Chakerian: Computer Attack Graph Generation Tool. Proceedings of DARPA Information Survivability Conference & Exposition II, June 2001.
18. P. Ryan, S. Schneider: Modeling and Analysis of Security Protocols - A CSP Approach. Addison-Wesley, 2001.
19. G. Rohrmair, G. Lowe: Using Data-Independence in the Analysis of Intrusion Detection Systems. Workshop on Issues in the Theory of Security (WITS'03), Warsaw, Poland, April 2003.
20. S. Jajodia, S. Noel, B. O'Berry: Topological Analysis of Network Attack Vulnerability. Managing Cyber Threats: Issues, Approaches and Challenges. V. Kumar, J. Srivastava, A. Lazarevic (eds.), Kluwer Academic Publisher, 2003.
21. S. Noel, S. Jajodia: Managing Attack Graph Complexity through Visual Hierarchical Aggregation. Proceedings of the ACM CCS Workshop on Visualization and Data Mining for Computer Security, Fairfax, Virginia, October 2004.
22. J. S. Shapiro: The Practical Application of a Decidable Access Control Model. Technical Report SRL-2003-04, John Hopkins University, 2003.
23. SANS Research Center, [online]: The SANS Top 20 Internet Security Vulnerabilities. Available from: <http://www.sans.org/top20/>.
24. J. R. Lipton, and L. Snyder: A Linear Time Algorithm for Deciding Subject Security. J. ACM. 24, 3 (Jul. 1977), 455-464.

# Multiplex Encryption: A Practical Approach to Encrypting Multi-recipient Emails

Wei Wei<sup>1</sup>, Xuhua Ding<sup>2</sup>, and Kefei Chen<sup>1</sup>

<sup>1</sup> Computer Science Department, Shanghai JiaoTong University  
{[www\\_weiwei](mailto:www_weiwei@sjtu.edu.cn), [kfchen](mailto:kfchen@sjtu.edu.cn)}@sjtu.edu.cn

<sup>2</sup> School of Information Systems, Singapore Management University  
[xhding@smu.edu.sg](mailto:xhding@smu.edu.sg)

**Abstract.** Efficiently protecting the privacy of multi-recipient emails is not as trivial as it seems. The approach proposed by S/MIME is to concatenate all ciphertexts. However, it suffers from poor scalability due to its linear computation and communication cost. In this paper, we propose a new practical and secure scheme, called *multiplex encryption*. By combining the ideas of identity-based mediated RSA and re-encryption mix network, our framework only requires constant computation and communication cost to encrypt a multi-recipient email.

## 1 Introduction

The electronic mail is one of the most popular media for data exchange. People send and read emails from their computers, PDAs or even cellular phones. A huge volume of information are transferred via emails within an organization or across organizations. Among them, those with confidential data are encrypted such that only the intended receivers are able to access the content while an adversary obtains no information about the protected data. With the support of the public key infrastructure, those emails are encrypted under the recipients' public keys. According to S/MIME[14], the current standard for secure emails proposed by IETF's S/MIME Working Group, a confidential email is encapsulated by an "encryption envelop". Specifically, the content is encrypted by a randomly selected symmetric key, which is encrypted under the recipient's public key.

It seems trivial to extend the one-to-one email encryption method to multi-recipient emails, in which case the same email is sent to a number of receivers by using *carbon-copy*. In S/MIME approach, the sender produces different encryption envelops for different recipients. All envelops are pushed into a stack. The receiver seeks and decrypts the envelop intended for him from the stack. Unfortunately, this approach has several drawbacks. First, the computation cost at the sender's end is linear to the number of recipients. Note that users usually expect immediate email forwarding. Therefore, the delay incurred by encryption offsets the user friendliness of emails, since it stalks email delivery. Secondly, the length of the message to transfer is linear to the number of recipients, which contradicts the motivation of using carbon-copy. In a SMTP transaction for

carbon-copy, the sender first interacts with the SMTP server to confirm every recipient’s address in the carbon copy list. Then, only one copy of email, instead of multiple copies, is forwarded to the SMTP server. Although the same protocol applies to sending an encrypted multi-recipient email, the message to forward is essentially a concatenation of multiple ciphertexts. Therefore, the communication cost is not saved. Lastly, not the least, the concatenation of ciphertext ruins an important feature of carbon copy. Carbon-copy is used not only for the purpose of cost saving, but usually as a means to imply that the same message is equally read by all the intended recipients. For instance, a sales representative sends carbon-copy of a draft of contract to both his supervisor and his clients, so that all related entities receive consistent information about the contract terms; a committee chair expresses his opinion to all committee members by sending a carbon-copy email so that everyone on the committee receive the same statement. We observe that this feature is not preserved when carbon-copy emails are encrypted. This is due to the fact that each recipient only decrypts his own part and has no knowledge of the results from others’ decryption.

Encryption of multi-recipient emails is in fact surprisingly challenging. Though the communication model is similar to multicast, the approach to multicast security is infeasible for emails. Multicast encryption requires stable group membership and a key establishment phase. For emails, the recipients are chosen when the sender composes the email. Therefore, the grouping of recipients is ephemeral. Furthermore, it is impractical to require all recipients to be online to agree on a key. An ideal solution would be to design a new public key encryption scheme such that the sender constructs an encryption key  $\mathcal{PK}$  from a set of public keys  $\{PK_0, \dots, PK_n\}$ . The ciphertext produced by  $\mathcal{PK}$  can be decrypted by corresponding private keys  $\{SK_0, \dots, SK_n\}$  while the notion of semantic security still preserves. However, it is an open question whether such an encryption scheme exists.

In this paper we take a systematic approach to encrypting multi-recipient emails. We present a *multiplex encryption* system for multi-recipient emails. Built on top of mediated RSA[5], our scheme enables the sender to multiplex several ciphertexts into one so that the encryption cost and the length of ciphertext is independent of the number of recipients. In the recipient’s end, our scheme introduces a partially trusted server which, functioning as *demultiplexor*, translates the ciphertext and forwards the results to the corresponding recipients respectively. Note that the server is not *fully* trusted in the sense that the server is unable to decrypt the ciphertext by itself.

The rest of the paper is organized as follows. In next section, we show the related work. The details of multiplex encryption are presented in Section 3. We discuss its security in Section 4 and its performance in Section 5. A summary is provided in Section 6, which concludes the paper.

## 2 Related Work

Our scheme is built on top of identity-based mediated RSA[5]. In [5], a user’s RSA public key is derived from its identity, e.g. an email address, and the corre-

sponding RSA private key is split into two shares. One share is secretly owned by the user and the other is secretly owned by a partially trusted server. To produce a signature or decrypt a ciphertext, the user has to collaborate with the server. The main motivation of their work is for fine-grained control over users' security capabilities.

Public key encryption in multi-user setting was studied in [1], which discussed the security relation between single-user setting and multi-user setting. In [9], Kurosawa shorten the ciphertext by a half for ElGamal and Cramer-Shoup encryption in multi-user setting. Bellare et. al. further investigated the security of random reuse in [2] to provide a general test of extending standard encryption schemes for multiple users.

The function of our de-multiplexer is similar to re-encryption mix servers[7] in terms of computation model. In a re-encryption mix network, the mix server transforms a ciphertext into another one in order to provide source and destination anonymity against traffic analysis.

### 3 An Multiplex Encryption System

#### 3.1 Architecture

The multiplex encryption system comprises three types of entities: a Certification Authority (CA), an encryption de-multiplexer, denoted by  $\mathcal{DM}$ , and a group of email users denoted by  $U_1, U_2, \dots, U_n$ . The CA governs the other two types of participants and generates keys for  $U_1, \dots, U_n$ . All the users are in the same email domain, which is served by  $\mathcal{DM}$ . Practically, a de-multiplexer can be a customized SMTP or POP3 server, where users retrieve their emails. Specifically,  $\mathcal{DM}$  plays the role of both mail delivery agent and security mediator as in [5].  $\mathcal{DM}$  is *partially* trusted, in the sense that it is trusted to execute the protocol honestly and does not collude with any malicious users. We observe that establishing a *fully* trusted party would resolve the problem of multi-recipient encryption in a trivial fashion. However, a fully trusted party (TTP) is usually unrealistic or undesirable due to its high security risk, i.e. compromising TTP alone will expose all users' secrets. We stress that  $\mathcal{DM}$  is not a TTP since a compromised or malicious  $\mathcal{DM}$  is not able to decrypt any user's encrypted emails.

Similar to other public key encryption schemes, our construction has three algorithms KeyGen, Enc and Dec, as shown in following sections.

#### 3.2 Key Generation

CA initializes the system parameters and generates RSA keys for all users.

First, CA chooses two 512-bit random safe primes  $p$  and  $q$  such that  $p = 2p' + 1$  and  $q = 2q' + 1$ , where  $p', q'$  are also primes. It sets  $N = pq$  and  $e = 3$  or 65537. CA then selects a collision resistant and division intractable[8] hash function  $\mathcal{H}()$ , for instance SHA-256.  $p, q, p'q'$  are kept secret while  $(N, e)$  and  $\mathcal{H}$  are public.

For  $1 \leq i \leq n$ , user  $U_i$ 's public key  $e_i$  is derived from its email address  $ID_i$ , exactly in the same fashion as in [5]. Specifically,

$$e_i = \mathcal{H}(ID_i) \parallel 0 \cdots 01,$$

where "||" denotes concatenation of two binary strings. Note that the least significant bit is set to 1 to ensure that  $e_i$  is odd and has overwhelming probability of being co-prime to  $\phi(N)$ . CA generates  $d_{s,i}$  and  $d_{u,i}$  such that  $ee_i d_{s,i} d_{u,i} = 1 \pmod{\phi(N)}$ . Then  $d_{u,i}$  is securely delivered to  $U_i$  while  $d_{s,i}$  is securely transferred to  $\mathcal{DM}$ . The details are shown in Figure 1.

**Algorithm KeyGen: Generating keys for  $U_i$  (executed by CA).**  
 Let  $t$  be a public security parameter.

1.  $s \leftarrow t - |\mathcal{H}()| - 1$
2.  $e_i \leftarrow \mathcal{H}(ID_i) \parallel 0^s \parallel 1$ , where  $0^s$  denotes a  $s$  bit binary string of 0-s.
3.  $d_{s,i} \xleftarrow{r} \mathbb{Z}_{\phi(N)}^*$ , i.e.  $d_{s,i}$  is a random number in  $\mathbb{Z}_{\phi(N)}^*$ .
4.  $d_{u,i} \leftarrow (ee_i d_{s,i})^{-1} \pmod{\phi(N)}$
5.  $d_{u,i}$  and  $d_{s,i}$  are securely distributed to  $U_i$  and  $\mathcal{DM}$  respectively.

**Fig. 1.** User Key Generation Algorithm

Let  $\mathcal{DM}$  choose for himself a public key  $pk$  and a private key  $sk$  for a public key scheme which is semantically secure under adaptive chosen ciphertext attack.  $\mathcal{DM}$ 's encryption setting is independent of the users' RSA setting. An encryption on message  $m$  using  $pk$  is denoted by  $\mathcal{E}_{pk}(m)$  while a decryption on ciphertext  $c$  using  $sk$  is denoted by  $\mathcal{D}_{sk}(c)$ .

### 3.3 Multi-recipient Email Encryption

When an email is only delivered to one recipient, its encryption is exactly the same as in identity-based mRSA[5]. The sender derives the recipient's public key from his email address as in Figure 1; then encrypts the email using a normal RSA encryption defined in PKCS#1[12]. Note that the sender does not need to load the recipient's public key certificate by virtue of identity-based RSA encryption.

To encrypt a multi-recipient email, the sender executes the Algorithm Enc shown in Figure 2.

It is optional for  $U_i$  to sign the email using his private key, depending upon whether message integrity is in consideration.  $U_i$  specifies all intended recipients' email address in his carbon-copy list as well as in the ciphertext  $C'$ . One ciphertext is sent to all recipients.

### 3.4 Multi-recipient Email Decryption

When a recipients,  $U_j$ , comes to fetch the mail  $C'$ ,  $\mathcal{DM}$  helps him to decrypt the message. The basic idea is to combine the concept of identity-based mediated

**Algorithm Enc:** Encrypting a multi-recipient mail  $m$  for user  $U_1, \dots, U_k$   
(executed by the sender)

1. Employ  $(e, N)$  as an RSA public key and encrypt the mail using RSA with OAEP padding[4,6] as defined in PKCS#1:  
 $C \leftarrow \text{OAEP-ENC}(m)^e \bmod N$
2. Encrypt  $C$  using  $\mathcal{DM}$ 's public key:  
 $C' \leftarrow \mathcal{E}_{pk}(C \| ID_1 \| \dots \| ID_k)$ .
3. Prepare S/MIME headers and send  $C'$  to  $\mathcal{DM}$  with  $U_1, \dots, U_k$  being on the carbon-copy list.

**Fig. 2.** Multi-recipient encryption algorithm

RSA[5] and re-encryption mix server.  $\mathcal{DM}$  first decrypts  $C'$  into  $C$  which is the RSA encryption of  $m$  under the public key  $(e, N)$ . For user  $U_j$ ,  $1 \leq j \leq k$ ,  $\mathcal{DM}$  translates  $C$  into the ciphertext corresponding to  $U_j$  without knowing  $m$ . The details of the decryption algorithm Dec for  $U_j$  and  $\mathcal{DM}$  are shown in Figure 3.

Note that OAEP decoding is involved in  $U_j$ 's step.  $\mathcal{DM}$  should not, and actually is unable to, execute OAEP decoding since  $\hat{C}$  is still a ciphertext.

In Figure 4, we present a conceptual comparison between our multiplex encryption protocol and S/MIME's approach to multi-recipient encryption. In

**Algorithm Dec:** Decrypting a multi-recipient mail  $C'$  for  $U_1, \dots, U_k$

Mail Server:

1. Decrypt  $C'$  into  $C$  by computing:

$$C \| ID_{r_1} \dots \| ID_{r'_k} = \mathcal{D}_{sk}(C')$$

2. Construct a recipient set  $\mathcal{R} = \{ID_{r_1}, \dots, ID_{r'_k}\}$  for email  $C'$ . If  $\mathcal{R}$  does not match the carbon-copy list in the SMTP header, abort. Otherwise,
3. On receiving  $U_j$ 's request for retrieving email  $C'$ , check if  $ID_j \in \mathcal{R}$ . If not, the request is rejected. Otherwise,
4. Compute  $U_j$ 's public key  $e_j$  by computing  
 $e_j = \mathcal{H}(ID_j \| 0 \dots 01)$
5. Translate  $C$  for  $U_j$  by computing  
 $\hat{C} = C^{e_j d_{s,j}} \bmod N$
6. Send  $\langle C', \hat{C}, ID_{r_1} \dots \| ID_{r'_k} \rangle$  to  $U_j$ .

User  $U_j$ :

1. Decrypt  $\hat{C}$  by computing  
 $\hat{m} = \hat{C}^{d_{u,j}} \bmod N$
2. Decode the OAEP encoding of  $\hat{m}$  to get message  $m$ .  
 $m \leftarrow \text{OAEP-DEC}(\hat{m})$

**Fig. 3.** Multi-recipient decryption algorithm

our scheme, the sender  $U_i$  only sends one ciphertext to  $k$  recipients while in S/MIME’s approach, the sender has to send a concatenation of  $k$  ciphertexts. In our approach, only two encryptions are needed, while in S/MIME approach,  $k$  public key encryptions are required.

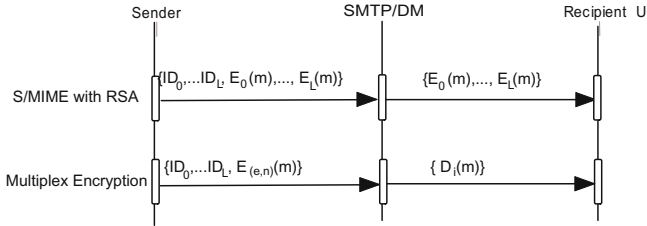


Fig. 4. Comparison Between Multiplex Encryption and Concatenated Encryptions

### 4 Security Discussion

Now we proceed to analyze the security of our multi-recipient encryption system. From the cryptography perspective, our construction is a variant of identity-based mediated RSA. In [5], the authors discussed several security issues, including the semantic security, the issue of public key generation, the issue of sharing RSA modulus etc. Their observations remain valid for our construction. Both their scheme and our encryption system share the same notion of semantic security. Besides those, we further discuss several security issues at the system level.

#### Malicious Users

For an email  $m$  intended for a set of users,  $\mathcal{R} = \{U_0, \dots, U_t\}$ , we consider whether a malicious user Eve,  $Eve \notin \mathcal{R}$ , is able to retrieve  $m$ . Note that if Eve knows the ciphertext  $C = m^e \bmod N$ , she can trivially decrypt it by computing  $C' = \mathcal{E}_{pk}(C || Eve)$  and sending  $C'$  to  $\mathcal{DM}$ . However, Eve is unable to obtain  $C$  since  $C$  is encrypted under  $\mathcal{DM}$ 's public key during email forwarding. Eve can not successfully mount a replay attack either, because the identities of intended recipients are encrypted together with  $C$  using  $\mathcal{E}_{pk}$ . Note that  $\mathcal{E}_{pk}(\cdot)$  is semantically secure under CCA2. According to [3], it is non-malleable under CCA2 as well. Therefore, she is unable to construct a ciphertext containing both her identity and message  $m$  without prior knowledge of  $m$ .

It is reasonable to assume that the channels between users and  $\mathcal{DM}$  are authentic, as all email retrieval protocols require user authentication, e.g. asking for user id and password. Therefore, it is infeasible for Eve to impersonate another user in  $\mathcal{R}$  to retrieve  $C$ . The message returned from  $\mathcal{DM}$  in Figure 3 is partially decrypted. However, since  $d_s^i$  is a random number chosen from  $\mathbb{Z}_{\phi(N)}^*$ , Eve gets no information about  $m$ .

## Malicious De-multiplexer

A malicious  $\mathcal{DM}$  may attempt to compromise the secrecy of an encrypted multi-recipient email.  $\mathcal{DM}$  only has knowledge of a set of random numbers  $d_{s,0}, d_{s,1}, \dots, d_{s,n}$ , which is exactly the same as in ID-based mediated RSA[5]. Given a ciphertext  $C$ ,  $\mathcal{GM}$  is unable to obtain any information on  $m$  from  $C^{d_{s,i}} \bmod N$  because he is not capable to distinguish the distribution of  $c^{d_{s,i}} \bmod N$  and  $c^r \bmod N$  where  $r$  is a random number. Therefore, the presence of a malicious  $\mathcal{DM}$  alone would not place a threat to the semantic security of our encryption.

## Collusion Between De-multiplexer and Users

As in [5], our construction is threatened by the collusion attack between a dishonest de-multiplexer and a malicious user. The collusion will destroy the whole system's security since they can collaboratively recover the factorization of  $N$ . In practice, an organization may alleviate the problem by further splitting  $d_{s,i}$  to several parts or by deploying a proactive threshold RSA scheme so that it is more challenging for the adversary to compromise several servers at the same time.

## Implication of Carbon Copy

In Section 1, we argue that S/MIME's approach does not preserve the consistency implication of carbon-copy since all recipients essentially access different ciphertexts.

With multiplex encryption, this feature of email carbon copy is saved because the recipients decrypt a common ciphertext, though in their own ways. It is in the same fashion as an email in plaintext sent to multiple recipients. When two recipients  $U_0, U_1$  result in different messages  $m_0, m_1$  for the same encrypted email  $C'$ , the discrepancy can be resolved in a court following the steps below:

1. Given  $C'$ ,  $\mathcal{DM}$  presents  $C$  and the randomness used in the encryption, which verifies that  $C$  is the plaintext of  $C'$  under  $\mathcal{DM}$ 's public key.
2.  $\mathcal{DM}$  presents  $\hat{C}_0 = C^{ed_{s,0}} \bmod N$  and  $\hat{C}_1 = C^{ed_{s,1}} \bmod N$ .
3.  $U_0$  runs RSA decryption on  $\hat{C}_0$  using their private keys and obtains the random seed  $s_0$  used for OAEP.  $U_1$  does the same on  $\hat{C}_1$  and obtains his random seed  $s_1$ . Both  $s_0$  and  $s_1$  are presented to the arbitrator.
4. The arbitrator verifies whether  $C$  is the RSA-OAEP encryption of  $m_0$  with random seed  $s_0$  or of  $m_1$  with seed  $s_1$ , under public key  $e$ . Note that only one of them is the valid plaintext and random seed pair.

## 5 Performance

### 5.1 Implementation

To validate our algorithms and evaluate the performance, we implemented a prototype of the multiplex encryption system by using OpenSSL[10]. It includes



1. CA and Admin Utilities: This is for certificate issuance and revocation. It is implemented on a Linux platform.
2. De-multiplexer  $\mathcal{DM}$ : It includes the general functions of SMTP and POP3 protocols. It receives an encrypted email for multi-recipients. In email retrieval phase, it transforms the encrypted email properly for the requesting client.  $\mathcal{DM}$  is implemented as a daemon running on a Linux host.
3. Outlook[11] plug-in: This is implemented as a Windows Dynamic Link Library(DLL) which provides encryption and decryption function for Outlook users.

A screen snapshot of the Outlook 2003 plug-in is shown in Fig. 5.

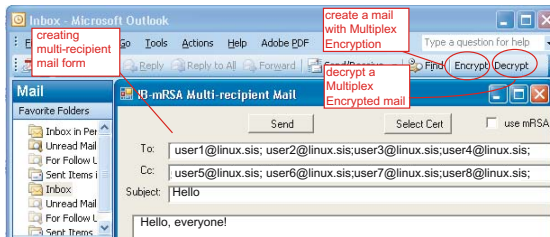


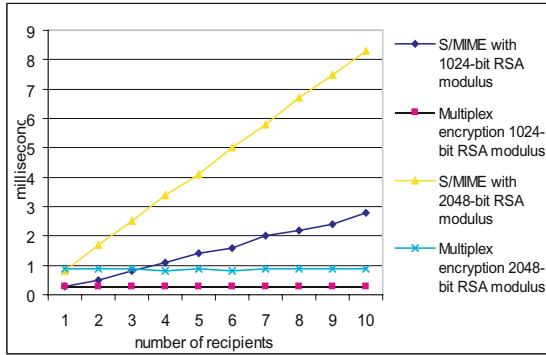
Fig. 5. Outlook 2003 Plug-in

## 5.2 Performance Analysis

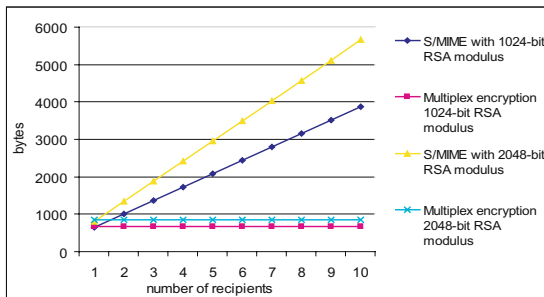
We conducted a series of experiments to evaluate the computation and communication cost of our scheme.  $\mathcal{DM}$  daemon was running on a Linux PC with an Intel Pentium IV 2.00GHz processor. The client was on a Windows PC with 1.6GHz Centrino processor. Both of them had 1GByte memory. We ran two groups of experiments: one with 1024bit RSA modulus and the other with 2048bit RSA modulus. In each group of experiments, we measured the performance for 1 to 10 recipients, respectively.

Figure 6 shows the RSA encryption time cost in milliseconds for the sender to encrypt an email for a number of recipients using two multi-recipient encryption approaches. The X axis indicates the number of recipients and the Y axis is for the encryption time. Not surprisingly, the two lines for S/MIME approach rise up linearly with respect to the number of recipients. In contrast, the encryption time using multiplex encryption almost remains unchanged though the number of recipients increases. It is straightforward to conclude from the protocol description in Figure 2 that the computation cost is independent of the number of recipients.

In Figure 7, we compare the communication cost for encrypted multi-recipient email delivery using two encryption approaches. We measure the cost in terms of the number of bytes to send by the sender. The X axis denotes the number of the recipients and the Y axis denotes the data length in bytes. Since the ciphertexts of each recipient are concatenated together in S/MIME approach, its data length



**Fig. 6.** Computation Cost for Multi-recipient Email Encryption



**Fig. 7.** Communication Cost for Encrypted Multi-recipient Emails

grows linearly with the number of recipients, as shown by the two rising lines in Figure 7. In contrast, only one ciphertext is sent in multiplex encryption, where the length is determined by the RSA modulus used in encryption, and therefore is almost constant.

Nonetheless, the benefit of our scheme is at the cost of additional computation load at the both the server end and the recipients' sides. For every recipient,  $\mathcal{DM}$  needs to perform one RSA decryption operation. Inevitably, a recipient performs one RSA decryption as well. However, since neither  $\mathcal{DM}$  nor the recipient knows the factorization of  $N$ , they are not able to take the advantage of Chinese Remainder Theorem (CRT), which usually facilitates RSA decryption three times faster when used in standard RSA settings. The partial decryption time for  $\mathcal{DM}$  and a recipient are shown in Table 1. We also list the standard RSA decryption time (with CRT) on the same hosts to demonstrate the additional cost.

We argue that the cost at both  $\mathcal{DM}$  and the recipient sides is deserved. First, when the number of recipient is large, the benefit from the sender side compensates the cost. Second, users usually expect immediate email delivery. Minimizing the delivery delay keeps the user-friendliness of email application. On the other hand, the email retrieval takes place periodically and is not a

**Table 1.** Decryption Time Cost for  $\mathcal{DM}$  and a Recipient (in ms)

RSA Modular size (bit)	$\mathcal{DM}$	Recipient	Standard RSA Decryption ( $\mathcal{DM}$ )	Standard RSA Decryption (User)
1024	13	13	5	4
2048	83	79	27	26

real-time application. Therefore, slightly longer delay in retrieval does not affect the recipients. Moreover, to improve the performance, the partial decryption at  $\mathcal{DM}$  end can be computed during idle time, instead of being triggered by the recipient's request.

## 6 Summary

Protecting the privacy of multi-recipient emails is not trivial as it seems. The approach proposed by S/MIME suffers from poor scalability due to its linear computation and communication cost. In this paper, we propose a new practical and secure scheme, called *multiplex encryption*, by combining the ideas of identity-based mediated RSA and re-encryption mixnet. Our framework only requires constant number of RSA encryption operations and constant ciphertext length for multi-recipient email encryption. We also implemented a prototype for experiment purpose.

Our scheme has a few drawbacks. First, it introduces a partially trusted third party. Though compatible with the email application architecture, deploying of our scheme still requires changes on email servers. Moreover, it limits all users to be in the same email domain. Secondly, the security relies on an assumption that  $\mathcal{DM}$  does not collude with any users. For a large, heterogenous organization, such assumption might not hold. Our future work will investigate the feasibility of a new encryption paradigm without the involvement of a third party.

## References

1. M. Bellare, A. Boldyreva and S. Micali: Public-key encryption in a multi-user setting: security proofs and improvements. In Eurocrypt'2000.
2. M. Bellare, A. Boldyreva and J. Staddon: Multi-recipient encryption schemes: security notions and randomness re-use. In PKC'2003.
3. M. Bellare, A. Sahai: Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-based Characterization. In Crypto'99
4. M. Bellare and P. Rogaway: Optimal asymmetric encryption – how to encrypt with RSA. In A.D. Santis, editor, *Advances in Cryptology - EUROCRYPT '94*
5. X. Ding, G. Tsudik: Simple identity-based cryptography with mediated RSA. In 2003 RSA Conference, *Cryptographers' Track*, April 2003.
6. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the rsa assumption. In *Advances in Cryptology - CRYPTO '2001*

7. P. Golle, M. Jakobsson, Ari Juels and P. Syverson: Universal Re-encryption for Mixnets. In *CT-RSA 2004*
8. R. Gennaro, S. Halevi, and T. Rabin: Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*
9. K. Kurosawa: Multi-recipient Public-Key Encryption with Shortened Ciphertext. In *5th International Workshop on Practice and Theory in Public Key Cryptosystems*, PKC 2002.
10. OpenSSL User Group. The OpenSSL Project Web Page, <http://www.openssl.org/>
11. Microsoft. Microsoft Outlook©, <http://www.microsoft.com>
12. PKCS#1 v2.1: RSA Cryptography Standard. RSA Laboratories, June 2002
13. PKCS#12: Personal information exchange syntax. RSA Laboratories, 1999. Version 1.0.
14. S/MIME: Secure/Multipurpose Internet Mail Extensions. The S/MIME Working Group: <http://www.ietf.org/html.charters/smime-charter.html> .

# Secure Multicast Using Proxy Encryption\*

Yun-Peng Chiu, Chin-Laung Lei, and Chun-Ying Huang

Department of Electrical Engineering,  
National Taiwan University  
{frank, huangant}@fractal.ee.ntu.edu.tw, lei@cc.ee.ntu.edu.tw

**Abstract.** In a secure multicast communication environment, only valid members belong to the multicast group could decrypt the data. In many previous researches, there is one “group key” shared by all group members. However, this incurs the so-called “1 affects n problem,” that is, an action of one member affects the whole group. We believe this is the source of scalability problems. Moreover, from the administrative perspective, it is desired to confine the impacts of changing membership events in a local area. In this paper, we propose a new secure multicast architecture without using a group key. We exploit a cryptographic primitive “proxy encryption.” It allows routers to convert a ciphertext encrypted under a key to a ciphertext encrypted under another key, without revealing the secret key and the plaintext. By giving proper keys to intermediate routers, routers could provide separation between subgroups. Therefore the goals of scalability and containment are achieved.

**Keywords:** Secure multicast, multicast key management, cipher sequences, proxy encryption, ElGamal cryptosystem.

## 1 Introduction

Since the commence of multicast communications in the late 1980s [1], secure multicast communication have been frequently addressed in the literature. Rafaeli and Hutchison’s paper [2] provided a detailed survey on secure multicast.

Quite a few researches in this area made use of a group key, which is shared among all group members. The sender encrypts the multicast data using this group key, and all valid members use the same group key to decrypt. However, the existence of this group-wise key incurs the so-called “1 affects n problem” [3], which means an action of one member affects the whole group. More specifically, since the group key is known by all members, whenever a member joins or leaves the group, everyone remains in the group must acquire a new group key.

To build a practical secure multicast architecture, we focus on scalability and containment issues. Scalability means that the processing overhead of each security action should be minimized in terms of number of group members. Containment means that a security event occurs in one subgroup does not affect other subgroups.

---

\* This work was supported in part by Taiwan Information Security Center (TWISC), National Science Council under the grants NSC 94-3114-P-001-001-Y.

To aim at the above issues, we adopt two techniques. First, distribute the computation loads to intermediate routers. This makes the whole architecture scalable. And second, make a dependency between keying material and the topology of the multicast network. This dependency assures the containment of security exposures.

A naive method to provide containment is to decrypt and then encrypt again at intermediate routers [3]. This method requires fully trust to routers because routers have the ability to decrypt the plaintext, which is an undesirable feature.

In this paper, we propose a new secure multicast architecture for large and dynamic groups. Specifically, we focus on the one-to-many communication pattern. We exploit a cryptographic primitive “proxy encryption.” By using this primitive, a proxy (router) could convert the ciphertext for one person into the ciphertext for another person without revealing secret decryption keys or the plaintext. Therefore the goal of scalability and containment could be achieved.

The rest of this paper is organized as follows. Related works and the basic concept of proxy encryption are discussed in Section 2. Section 3 describes the proposed secure multicast architecture based on proxy encryption. We analyze the proposed scheme, and compare it with related works in Section 4. Finally, Section 5 concludes this paper.

## 2 Related Works

In this section, we discuss some previous researches. Logical Key Hierarchy (LKH) may be the most representative research in this area; many researches followed their methodology and tried to enhance it. The cipher sequences framework (CS) tries to solve the multicast security problem using a different methodology. The most important advantage of CS is the containment. We also discuss Mukherjee and Atwood’s researches, which also make use of proxy encryption.

### 2.1 Logical Key Hierarchy

Logical Key Hierarchy (LKH) is separately proposed by Wallner et al. [4] and Wong et al. [5]. In this approach, all group members form a “logical” tree. The root node represents the group key shared by all group members, the leaf nodes are members, and each inner node represents a key encryption key (KEK). Besides the group key, a member also has a set of KEKs, including the KEK of each node in the path from its parent to the root. For example, in Fig. 1, member  $u_5$  will have  $k_5$ ,  $k_{56}$ ,  $k_{58}$ , and the group key,  $k$ . Therefore, in a balanced tree, a member will have  $(\log_2 \mathcal{N}) + 1$  keys, where  $\mathcal{N}$  is the group size, and  $\log_2 \mathcal{N}$  is the height of the tree. When a rekeying is needed, these KEKs could be used to encrypt new KEKs. For example, if member  $u_5$  leaves the group, we must change those keys it knows. Therefore new KEKs  $k'_5$ ,  $k'_{56}$ ,  $k'_{58}$  and the new group key  $k'$  are generated. These new keys are encrypted using KEKs and transmitted to remaining members. We encrypt new  $k'_{56}$  using  $k_6$ , and encrypt new  $k'_{58}$  using  $k'_{56}$  and  $k_{78}$ , respectively. Then  $k'$  is encrypted using  $k'_{58}$  and  $k_{14}$ , respectively.

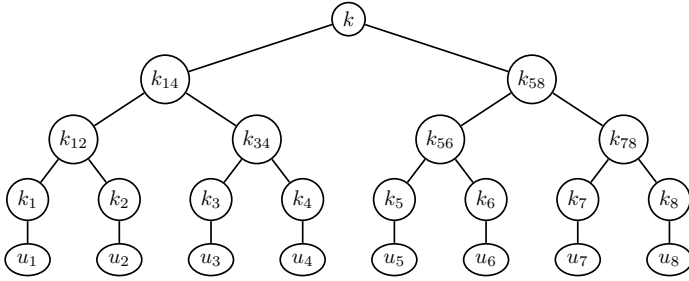


Fig. 1. An LKH tree

Finally these encrypted keys are multicast to the whole group. All remaining members could get new KEKs and the group key from these encrypted keys.

### 2.2 Cipher Sequences

The cipher sequences framework (CS) was proposed by Molva and Pannetrat [6]. By distributing secure functions to intermediate nodes, keying material has a dependency on the multicast network topology. Therefore the containment of security exposures is assured.

Assume  $S_0$  is the information to be multicast. Each node  $N_i$  is assigned a secret function  $f_i$ .  $N_i$  receives multicast data from its parent node  $N_j$ , computes  $S_i = f_i(S_j)$ , and forwards the result  $S_i$  to its children. A leaf eventually receives  $S_n^i$ . Each leaf is given a reversing function  $h_i$ , and it can use  $h_i$  to get the original multicast data by calculating:  $S_0 = h_i(S_n^i)$ .

For example, Fig. 2 depicts a simple tree with five cipher sequences. We follow one cipher sequence from the root to the leaf  $M_5$ . First, the root computes  $f_1(S_0)$  and sends the result to its children inner nodes.  $R_2$  receives  $S_1^4 = f_1(S_0)$ , computes and sends  $f_5(S_1^4)$  to  $R_5$ . Then  $R_5$  receives  $S_2^4 = f_5(S_1^4)$  and sends  $f_6(S_2^4)$  to the leaf  $M_5$ . Finally, the leaf  $M_4$  receives  $S_3^4 = f_6(S_2^4)$  and recovers the original multicast data by computing  $S_0 = h_4(S_3^4)$ .

### 2.3 Proxy Encryption

Proxy encryption was first introduced by Blaze, Bleumer, and Strauss in 1998 [7]. The basic idea is that a proxy, given a proxy key, could convert the ciphertext for one person into the ciphertext for another person without revealing the secret decryption keys or the plaintext.

In traditional public-key encryption schemes, a message  $m$  encrypted using  $A$ 's public key  $EK_A$  could only be decrypted using  $A$ 's private key  $DK_A$ . On the contrary, in a proxy encryption scheme, a new role, proxy  $P$ , is introduced.  $P$  is given a proxy key  $\pi_{A \rightarrow B}$ , and  $P$  could convert a ciphertext originally for user  $A$  to a message which could be decrypted using user  $B$ 's private key.  $P$  could not decrypt the ciphertext, nor gain information about  $A$ 's or  $B$ 's private keys.

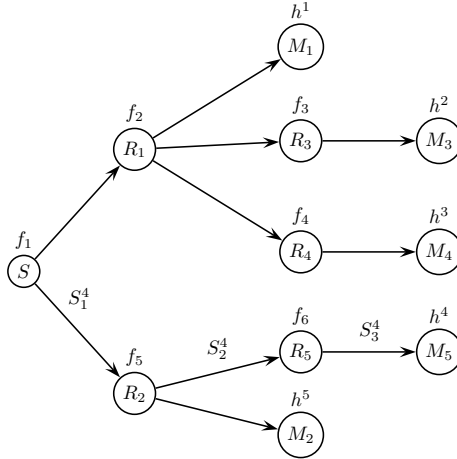


Fig. 2. An example of CS tree

In this paper, we use the “unidirectional ElGamal encryption scheme” proposed by Ivan and Dodis [8] as an example of underlying proxy encryption scheme in our architecture. Our architecture is not limited to it, and other proxy encryption schemes could also be used.

Assume an ElGamal encryption scheme is defined as a three-tuple: (KeyGen, Enc, Dec). The key generation algorithm KeyGen outputs a public key  $EK = (g, p, q, y)$  and a private key  $DK = (x)$ , where  $q$  is a prime number,  $p$  is a prime number of the form  $2q + 1$ ,  $g$  is a generator in  $\mathbb{Z}_p^*$ ,  $x$  is randomly chosen from  $\mathbb{Z}_q$ , and  $y = g^x \pmod p$ . The encryption algorithm Enc is defined as  $c = \text{Enc}(m, EK) = (g^r \pmod p, mg^{xr} \pmod p)$ , where  $r$  is randomly chosen from  $\mathbb{Z}_q$ .  $r$  is chosen by the sender, and  $r$  should be used only once. The decryption algorithm Dec decrypts the ciphertext by computing  $mg^{xr}/(g^r)^x = m \pmod p$ .

The unidirectional ElGamal encryption scheme proposed in [8] could be defined as (KeyGen, Enc, Dec, ProxyKeyGen, ProxyEnc). The key generation algorithm KeyGen outputs a public/private key pair as the original ElGamal encryption scheme. The encryption and decryption algorithms are also the same with the original version. ProxyKeyGen splits  $DK_A = x$  into two parts  $x_1$  and  $x_2$ , where  $x = x_1 + x_2$ . Then  $x_1$  is given to the proxy, and  $x_2$  is given to user  $B$ . Using ProxyEnc,  $P$  could convert a message originally for user  $A$  to a message for user  $B$ . ProxyEnc has two parameters:  $c$  and  $\pi_{A \rightarrow B}$ , here  $c = \text{Enc}(m, EK_A)$ , and  $\pi_{A \rightarrow B} = x_1$ . On receiving  $c = (g^r \pmod p, mg^{xr} \pmod p)$ , the proxy computes  $mg^{xr}/(g^r)^{x_1} = mg^{(x-x_1)r} = mg^{x_2r}$ , and then sends  $(g^r \pmod p, mg^{x_2r} \pmod p)$  to user  $B$ . Finally, user  $B$  can decrypt this converted message as  $mg^{x_2r}/(g^r)^{x_2} = m$ .

### 2.4 Mukherjee and Atwood’s Researches

Mukherjee and Atwood proposed a key management scheme exploiting the proxy encryption technique in [9]. In their scheme, there are Group Manager (GM),



Group Controllers (GCs), and participants (members). When a group is created, a node is set up as the GM. The GM is configured with group and access control information, and it generates the encryption/decryption keys. In a multicast tree, there may be several GCs, and each GC is associated with a subtree of the distribution tree. GCs perform key management functions and transform the ciphertext using proxy encryption. Participants join the GC nearest to them, and get keys from the GC.

When a rekeying is required, the GC “splits” the group decryption key to the “proxy encryption key” and the “proxy decryption key.” For example, in the unidirectional ElGamal encryption scheme, the group decryption key  $x$  is split into  $x_1$  and  $x_2$ , such that  $x = x_1 + x_2$ . Then the GC applies transformations.

When a member joins, the GC sends the proxy decryption key to the joining participant over a secure channel protected using a shared key  $K_g$ , and multicasts the proxy decryption key to other members. When a member leaves, the GC sends the proxy decryption key to the remaining participants by: unicasting the proxy decryption key to each participant over separate secure channels, or, encrypting the proxy decryption key using each participant’s  $K_g$  and multicasting an aggregated message.

In this scheme, a group key is still used. Proxy encryption is used only between a membership change event and a periodic rekeying. After a periodical rekeying, a new group key is used and the transformation is stopped. On the contrary, in our scheme, proxy encryption is always used to transmit data. Moreover, in their scheme, intuitive methods are used to deliver the proxy decryption key to participants, which brings a large burden to GCs.

### 3 The Proposed Scheme

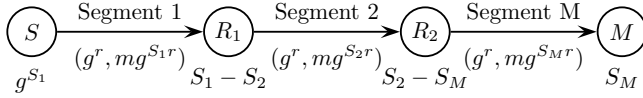
In this section, we propose a new secure multicast architecture making use of the proxy encryption mechanism discussed in the previous section. In the following subsections, we first extend the unidirectional ElGamal encryption to a proxy sequence. Then our multicast model is described in Section 3.2. Finally we extend the unidirectional ElGamal encryption to a multicast tree in Section 3.3.

#### 3.1 Extend to a Proxy Sequence

In this subsection, we extend the original unidirectional ElGamal encryption scheme to a case with a sequence of proxies, as shown in Fig. 3. The sender  $S$  sends the message along the path. Routers  $R_1$  and  $R_2$  play the role of proxies. The final destination is  $M$ .

Here we introduce a new concept “segment key.” Each link between two nodes is called a “segment.” Each segment is assigned a public/private key pair by a trusted server ( $TS$ ). These segment keys are actually stored in  $TS$ . Moreover,  $TS$  calculates proxy keys according to segment keys, and distributes proxy keys to intermediate routers by secure channels.

We show an example of using the unidirectional ElGamal encryption scheme in Fig. 3. All parameters are the same with those in the unidirectional ElGamal



**Fig. 3.** Proxy sequence using unidirectional ElGamal encryption

encryption scheme. We let the proxy key between Segment A and Segment B be the difference of their corresponding private key, i.e.,  $\pi_{A \rightarrow B} = A - B$ , where  $A$  and  $B$  are private keys of Segment A and Segment B, respectively. At first,  $S$  encrypts message  $m$  using Segment 1’s public key,  $g^{S_1}$ , and sends the ciphertext  $(g^r \bmod p, mg^{S_1 r} \bmod p)$  to  $R_1$ .  $TS$  gives  $R_1$  a proxy key,  $\pi_{S_1 \rightarrow S_2} = (S_1 - S_2)$ , therefore  $R_1$  computes  $mg^{S_1 r} / (g^r)^{(S_1 - S_2)} = mg^{S_2 r}$ , and then sends  $(g^r \bmod p, mg^{S_2 r} \bmod p)$  to  $R_2$ . Similarly,  $R_2$  has  $\pi_{S_2 \rightarrow S_M} = (S_2 - S_M)$ , so it converts the received ciphertext into  $(g^r \bmod p, mg^{S_M r} \bmod p)$ . Finally,  $M$  has  $S_M$ , therefore it could compute  $mg^{S_M r} / (g^r)^{S_M} = m$  to decrypt the ciphertext.

### 3.2 Multicast Model

Our multicast model is similar to that in the cipher sequences framework. We use Fig. 2 again to describe our multicast model. In a multicast routing protocol, routers form a multicast tree to transmit multicast traffic. The root  $S$  is the source, intermediate nodes  $R_i$ , where  $i$  is an integer, are routers, and every leaf node represents a set of local subgroup members attached to the same router.  $M_i$  is the set of local subgroup members attached to  $R_i$ . Each router may have local subgroup members and/or downstream routers. Note in the cipher sequence framework, intermediate routers do not have local members.

In LKH schemes, all keying operations are done by senders and members; routers are not involved. Instead, in our scheme, we make use of routers, because this reduces the loads of senders and members. The trade-off is that we must grant some trust to routers. We assume routers faithfully transfer and convert encrypted multicast data.

### 3.3 Extend to a Multicast Tree

Based on Section 3.1, now we further extend to a multicast tree, for example as shown in Fig. 4. We assume that  $TS$  knows the overall topology of the multicast tree. Although  $TS$  seems to be centralized, we think the tasks of  $TS$  could be easily distributed over several network entities. As mentioned,  $TS$  generates keys and distributes them to intermediate routers by secure channels.

We define a term “downstream-segment set,” which is a set includes segments among a given router and all its downstream routers. (Note the segment between a router and its local subgroup is not included in this set.) For instance, in Fig. 4, Segment 3 and Segment 4 are in the same downstream-segment set. In our scheme, segments belong to the same downstream-segment set are given the same segment key.

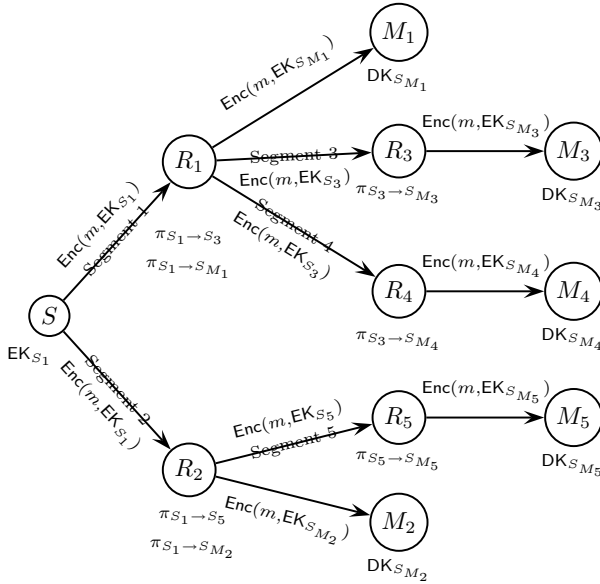


Fig. 4. Proxy tree

Each intermediate router is given two keys: the *downstream routers conversion key* and the *local subgroup conversion key*. The former is used to convert the upstream ciphertext to another ciphertext for downstream routers; it is calculated according to the upstream segment’s key and the downstream segment’s key. On the other hand, the latter is used to convert the upstream ciphertext for its local subgroup members only; it is calculated according to the upstream segment’s key and the local subgroup key.

Each local subgroup has its own local subgroup key. Members of local subgroup use this key to decrypt the ciphertext from the attached router. We assume each local subgroup uses its own secure multicast key distribution protocol. This permits local policy of choosing a key distribution protocol. Every subgroup could choose its own key distribution protocol independent from other subgroups. Since the size of a local subgroup is much smaller than that of the whole group, using LKH is acceptable in local subgroups.

We also assume every subgroup has a “subgroup controller,” who is responsible for local subgroup key management. It maintains logical keys for the local subgroup, and securely transmits the local subgroup key to *TS*. After members of local subgroup have determined their subgroup key, the subgroup controller securely transports their subgroup key to *TS*, then *TS* computes the proxy key for the attached router according to the local subgroup key.

The key assignment algorithm is described in Algorithm 1. We use Fig. 4 as an example. *S* encrypts the message *m* using Segment 1’s public key,  $EK_{S_1}$ , and sends the ciphertext  $Enc(m, S_1)$  to both  $R_1$  and  $R_2$ . We send the same encrypted message to  $R_1$  and  $R_2$ , and this could preserve the benefit of multicast: the same

---

**Algorithm 1:** Key assignment algorithm

---

**Input:** Multicast tree topology  
**Output:** Key assignment on every router  
**for** every segment except that between a router and its local subgroup **do**  
    **if** no segment in the same downstream-segment set is already given a key **then**  
        assign a segment key;  
    **end**  
**end**  
**for** every router **do**  
    **if** has downstream routers **then**  
        assign a downstream routers conversion key according to the upstream segment's key and the downstream segment's key;  
    **end**  
    **if** has local members **then**  
        assign a local subgroup conversion key according to the upstream segment's key and the local subgroup key;  
    **end**  
**end**

---

data is sent to different paths.  $R_1$  is given  $\pi_{S_1 \rightarrow S_3}$ , then it sends  $\text{Enc}(m, \text{EK}_{S_3})$  to  $R_3$  and  $R_4$ .  $R_1$  also has  $\pi_{S_1 \rightarrow S_{M_1}}$ , this proxy key allows  $R_1$  to convert the upstream ciphertext into  $\text{Enc}(m, \text{EK}_{S_{M_1}})$ , which could be decrypted only by using  $M_1$ 's local subgroup key,  $\text{DK}_{S_{M_1}}$ . Similarly,  $R_4$  is given  $\pi_{S_3 \rightarrow S_{M_4}}$ , therefore it converts  $\text{Enc}(m, \text{EK}_{S_3})$  to  $\text{Enc}(m, \text{EK}_{S_{M_4}})$ .  $M_4$  could use its local subgroup key  $\text{DK}_{M_4}$  to decrypt the message.

In this way, every intermediate router converts the upstream ciphertext into the downstream ciphertext or the local subgroup ciphertext. On each link, multicast data is encrypted, so it is infeasible for an eavesdropper with no proper keys to decrypt multicast data. Therefore, we achieve secure transmission of multicast traffic.

### 3.4 Rekey

When a member joins or leaves the group, the local subgroup key should be changed. At the same time, the router responsible for this subgroup should also change a new proxy key according to the new local subgroup key in order to convert ciphertext for local members use.

When there are downstream nodes/routers interesting to the group, a multicast router should connect itself (and its downstream routers) to the multicast tree of this group. On the other hand, when a router has no downstream nodes/routers belong to this group, this router will be removed from the tree. When a router joins/leaves the tree, related segment keys must be changed; routers related with these segment keys must also change new proxy keys. New keys are generated and distributed by  $TS$  securely.

In multicast security, forward/backward secrecy are usually discussed, we follow the definitions in [10]. Forward secrecy is that a passive adversary who knows a subset of old group keys cannot infer subsequent group keys. Backward secrecy is that a passive adversary who knows a subset of group keys cannot infer previous group keys.

---

**Algorithm 2:** Rekey when a router joins/leaves
 

---

**Input:** Multicast tree topology, the ID of joined/left router:  $R$ **Output:** New key assignment on related routerchange  $R$ 's upstream segment's key;

change the segment keys in the same downstream-segment set;

change the downstream routers conversion key of routers connected to those changed segments;

---

Algorithm 2 describes the rekeying algorithm when a router joins or leaves. Look Fig. 4 for example, if all members in  $M_4$  leave the group,  $R_4$  will be deleted from the tree. In order to achieve forward secrecy, the related segment key  $S_3$  should be changed to  $S'_3$ , and the related routers  $R_3$  and  $R_4$  must change their proxy keys.  $R_3$  and  $R_4$  get new proxy keys  $\pi_{S'_3 \rightarrow S_{M_3}}$  and  $\pi_{S'_3 \rightarrow S_{M_4}}$ , respectively.  $R_1$  must also change a new proxy key  $\pi_{S_1 \rightarrow S'_3}$ . On the other hand, if a new router wants to attach himself to  $R_2$ , in order to preserve backward secrecy, Segment 5 must be rekeyed. Thus  $R_2$ 's  $\pi_{S_1 \rightarrow S_5}$  and  $R_5$ 's  $\pi_{S_5 \rightarrow S_{M_5}}$  must be changed.

## 4 Analysis

In this section, first we examine the security of our scheme, and then we compare features and costs of related works and ours.

### 4.1 Security Analysis

By the definition of proxy encryption, it is infeasible for a proxy to derive traffic decryption keys with only proxy keys. Therefore intermediate routers could not decrypt the multicast data.

The local subgroup changes a new key when a member leaves/joins. When a member joins, the new member cannot infer previous local subgroup keys. Thus the new member cannot decrypt previous multicast data. On the other hand, when a member leaves, only remaining subgroup members get this new key. Thus the left member cannot decrypt subsequent multicast data. When a router leaves/joins, its upstream segment changes a new key. New keys are given to related routers by  $TS$  securely, and the left/new router cannot infer those keys. Therefore the router will not be able to transform subsequent/previous multicast data. As we can see, forward/backward secrecy is ensured in our protocol. Moreover, because every local subgroup is isolated with each other, members in the different subgroups gain no more information through collusion.

### 4.2 Comparisons of Features

The comparisons of features are shown in Table 1. In LKH, no intermediate nodes are involved, so no containment is provided. All other three schemes share computation loads to routers, and grant limited trust on routers. Thus containment is provided in these schemes.

**Table 1.** Comparisons of features

	Trust to intermediate nodes	Containment
LKH	No intermediate nodes	NO
CS	Limited trust	YES
Mukherjee	Limited trust	YES
Our scheme	Limited trust	YES

### 4.3 Comparisons of Costs

In this subsection, we analyze various costs of our scheme, and compare with related works. We assume the size of the whole group is  $\mathcal{N}$ , and the size of local subgroup is  $\mathcal{M}$ . Furthermore,  $\mathcal{N} \gg \mathcal{M}$ . The number of downstream routers connected to a router is  $\mathcal{P}$ . Assume we use LKH as our local key distribution protocol in our architecture.

In CS, every member needs a reversing function, and the sender and every intermediate node need a secret function. This seems efficient, but in fact, the trade-off is that the central server assigns every member a reversing function. Therefore the rekeying cost of the central server is  $O(\mathcal{M})$ .

In Mukherjee and Atwood's scheme, each member stores two keys. One is group decryption key or proxy decryption key; the other is the key shared with its GC,  $K_g$ . A GC shares  $K_g$  with each member, so it stores  $\mathcal{M}$  keys. And the sender needs one key.

In our scheme, a member requires  $(\log_2 \mathcal{M} + 1)$  keys. By contrast, a member in the original LKH scheme stores  $(\log_2 \mathcal{N} + 1)$  keys. Because LKH is only used in local subgroups, and  $\mathcal{N} \gg \mathcal{M}$ , members in our scheme store fewer keys than those in the original LKH schemes. Moreover, in our scheme, the sender only stores one encryption key, and intermediate routers only stores one conversion key. The result is shown in the left three columns of Table 2. As we can see, our scheme is efficient in storage.

**Table 2.** Comparisons of costs

	Member storage	Sender storage	Intermediate node storage	Rekeying cost
LKH	$\log_2 \mathcal{N} + 1$	$\mathcal{N}$	No intermediate nodes	$O(\log_2 \mathcal{N})$
CS	1	1	1	$O(\mathcal{M})$
Mukherjee	2	1	$\mathcal{M}$	$O(\mathcal{M})$
Our scheme	$\log_2 \mathcal{M} + 1$	1	1	$O(\log_2 \mathcal{M})$

In our scheme, because we use LKH as our local key distribution protocol, the cost to rekey a subgroup is  $O(\log_2 \mathcal{M})$ . When a router joins/leaves,  $TS$  must change  $\mathcal{P}$  segments keys, and then change  $\mathcal{P}$  proxy keys. Therefore the cost of rekeying when a router joins/leaves is  $O(\mathcal{P})$ . The result of comparison of rekeying cost is shown in the last column of Table 2. As we can see, our scheme is also efficient in rekeying.

## 5 Conclusions and Future Work

In this paper, we proposed a new secure multicast architecture. We eliminated the usage of the group key, because it is the source of scalability problems. We

exploited proxy encryption to allow intermediate routers to transform the ciphertext without revealing the secret key and the plaintext. By giving proper conversion keys to intermediate routers, the impacts of changing membership events are confined in a local area. Thus we achieved the goal of containment and scalability. Moreover, we have shown the rekeying procedure is secure and efficient. Therefore, our scheme is scalable for large and dynamic multicast groups.

Our architecture is not limited to the unidirectional ElGamal encryption scheme; other proxy encryption schemes could also be used. In the future, we will find more efficient cryptographic primitives suitable for our architecture. For example, the unidirectional identity-based encryption scheme proposed in [8] may be a good candidate. Currently, most proxy encryption schemes are for public-key cryptosystems, but there is also a research using symmetric ciphers [11]. Because of the efficiency of symmetric ciphers, that would also be a promising candidate.

## References

1. Deering, S.E., Cheriton, D.R.: Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems* **8** (1990) 85–110
2. Rafaeli, S., Hutchison, D.: A survey of key management for secure group communication. *ACM Computing Surveys* **35** (2003) 309–329
3. Mittra, S.: Iolus: A framework for scalable secure multicasting. In: *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*. (1997) 277–288
4. Wallner, D.M., Harder, E.J., Agee, R.C.: Key management for multicast: Issues and architectures. RFC 2627 (1999)
5. Wong, C.K., Gouda, M., Lam, S.S.: Secure group communications using key graphs. *IEEE/ACM Transactions on Networking* **8** (2000) 16–30
6. Molva, R., Pannetrat, A.: Scalable multicast security with dynamic recipient groups. *ACM Transactions on Information and System Security* **3** (2000) 136–160
7. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: *Proceedings of Advances in Cryptology - EUROCRYPT '98: International Conference on the Theory and Application of Cryptographic Techniques*. Volume 1403 of LNCS. (1998) 127–144
8. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: *Proceedings of the tenth Network and Distributed System Security Symposium*. (2003)
9. Mukherjee, R., Atwood, J.W.: Proxy encryptions for secure multicast key management. In: *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*. (2003) 377–384
10. Kim, Y., Perrig, A., Tsudik, G.: Simple and fault-tolerant key agreement for dynamic collaborative groups. In: *Proceedings of the 7th ACM conference on Computer and communications security*. (2000) 235–244
11. Cook, D.L., Keromytis, A.D.: Conversion and proxy functions for symmetric key ciphers. In: *Proceedings of the IEEE International Conference on Information Technology: Coding and Computing, Information and Security Track*. (2005) 662–667

# Efficient and Non-interactive Timed-Release Encryption

Julien Cathalo\*, Benoît Libert\*\*, and Jean-Jacques Quisquater

UCL Crypto Group,  
Place du Levant, 3. B-1348 Louvain-La-Neuve, Belgium  
{cathalo, libert, jjq}@dice.ucl.ac.be

**Abstract.** This paper revisits the important problem of sending a message “into the future” in such a way that no communication is needed between the server and other entities. This problem was recently re-investigated by Blake and Chan who showed a scalable non-interactive solution without considering a formal security model. We fill this gap by introducing a new stringent model tailored to the non-interactive setting. We then propose a new construction fitting our model and we show that it is more efficient than the recent non-interactive proposal (for which we also give a security proof in our model). We then explain how to provide our scheme and the one of Blake and Chan with an additional security property that strengthens the anonymity of receivers.

**Keywords:** timed-release encryption, formal models, provable security.

## 1 Introduction

The problem of sending a message “into the future”, i.e. encrypting a message so that its recipient cannot decrypt it prior to a pre-determined instant chosen by the sender, has been found to have many real-world applications such as electronic auctions, key escrow, scheduled payment methods, sealed-bid auctions, lotteries, etc.. It was first suggested by May [26] in 1993 and further studied by Rivest, Shamir and Wagner [29].

Two essential approaches have been investigated to solve the problem: the time-lock puzzle approach ([6,29,25,14,22,23]) and the trusted server approach ([17,26,29,12]). In the former, the receiver of an encrypted message has to invest in a significant computational effort to solve a reasonably small-size problem before obtaining the message. This approach does not involve a server but it turns out to be computationally expensive for the receiver and only solves the problem with approximately controllable release-times depending on the receiver’s computational power and on the moment at which the decryption operation is started. Sending a message that can be read at a precise moment (say 12:00am, July 31, 2005 GMT for example) turns out to be difficult using this approach.

---

\* This author’s work is supported by *Walloon Region / WIST-MAIS project*.

\*\* This author thanks the DG TRE’s *First Europe Program* of the *Walloon Region* and the *European Social Fund*.



On the other hand, in the trusted server approach, a trusted entity providing a common and absolute time reference is necessary to synchronize senders and receivers. Ideally, the server should have as few interactions as possible with senders and receivers. Up to very recently, the latter requirement was not satisfied by server-based solutions. In a system proposed by May ([26]), the server is an escrow agent storing messages and releasing them at specified times. Another method used by Rivest et al. [29] also requires interactions between the server and senders who must reveal their identity and their message's release-time.

In 1999, Di Crescenzo et al. ([17]) proposed a protocol, supported by a formal security model, and wherein senders are anonymous and do not have to interact with the server. Unfortunately, the latter has to engage in a conditional oblivious transfer protocol with receivers. As a result, the latter are not anonymous and the protocol is subject to denial-of-service attacks.

In 2001, when Boneh and Franklin published their famous identity based encryption (IBE) scheme ([12]), they also mentioned encryption "for the future" as a possible application. Their idea was to use identities augmented with release times as public keys. This solution is not scalable for small time granularities as the trusted private key generator has to deliver new private keys to each user at the start of each time period. Other IBE-based approaches ([16,27,11]) consider release-times as identities and trusted authorities as time servers issuing time-specific trapdoors at the beginning of each time period. These methods alone only allow the universal disclosure of encrypted documents.

Encrypting a message for a designated receiver and a specific future moment is possible by combining IBE-based unlock methods of [16,27,11] with a traditional public key encryption scheme. Such a composition is especially attractive with the time-based decryption procedure suggested by Boneh, Boyen and Goh ([11]) which consists in using the tree-like structure of Canetti et al. [15] backwards: indeed, it allows recovering past time-specific trapdoors from a current trapdoor. Nevertheless, the root of the tree-like structure of Canetti et al. ([15]) has to correspond to the last time period which implies an upper bound on the lifetime of the system. Unless special precautions are taken, such a composition would also leak information on the release-time of ciphertexts as the hierarchical IBE system of [11] does not have the receiver-anonymity property in the sense of Bellare et al. ([5]).

In this paper, we do not only focus on improving the efficiency of generic constructions. We also aim at providing TRE systems with the property of *release-time confidentiality* according to which ciphertexts do not reveal information to anyone but the intended receiver about their release-time. We also stress that the problem that we address is different from the 'token-controlled public key encryption problem' ([4]) where a sender encrypts a message using a specific token before handing it to a semi-trusted agent (that must communicate with senders who cannot remain anonymous) who stores it until it can be made available to the receiver for completing the decryption.

In our context, a scalable timed-release encryption (TRE) scheme wherein the time server never has to interact with the sender nor the receiver was recently

suggested by Blake and Chan ([8]) who were followed by [28,24] and for different applications by ([19]). In these settings, the sole responsibility of the server is to periodically issue time-specific trapdoors enabling the decryption of ciphertexts encrypted “for the future” ([8,28]) or the hatching of signatures ([19]).

It is to be noted that the scalable TRE solution given by Blake and Chan was not directly supported by a formal security model and only informal security arguments were given in [8] for a scheme that can also be thought of as a particular case of a solution proposed in [1] to tackle with access control problems using pairing based cryptography. We believe that security models considered in [1,24] should be strengthened a little and one of our contributions is to consider a more stringent formal security model for the specific application of non-interactive timed-release encryption. We have to mention that, independently of our work, [28] also considers a security model for authenticated timed-release encryption schemes. We focus here on the mere public key encryption case and we believe our model to be stronger than the one in [28] as well.

In this paper, we also propose a more efficient non-interactive TRE scheme than [8] and [1]. In anonymity enhancing purposes, we then explain how to avoid having to transmit release-times of ciphertexts in clear through insecure channels by hiding them from anyone but their intended recipient and we show how to add this security property to the scheme of [8] for which we also give a security proof in our model.

Both solutions may find other applications than the timed-decryption of digital documents. Similarly to the non-interactive solution of Blake and Chan, ours can be turned into an event-release encryption (ERE) scheme solving the problem of a sender who wishes to send a message that the recipient can only decrypt if a specific event occurs. As an example, we think of the context of a war-correspondent sealing an envelope containing sensitive information with the instruction “to open only if something happens to me”. In such a situation, the time server can be turned into a notary that has to verify the occurrence of the prescribed event before issuing a certificate testifying of the event’s happening.

Before describing our solutions, we formally define the concept of non-interactive timed-release encryption and we introduce a strong adversarial model which is inspired from the one of certificateless encryption schemes (CLE) ([2,3]). Section 3 then explains why a secure TRE scheme cannot be generically obtained from a secure CLE scheme contrary to what appears at first glance. The new TRE system is presented in section 4 while section 5 explains how to provide our system and the one of Blake and Chan with the newly defined release-time confidentiality property.

## 2 Formal Definition and Adversarial Models

Our model of timed-release encryption schemes assumes that ciphertexts always contain information about their release-time. More precisely, for some  $t \in \mathbb{N}$ , their last  $t$  bits are a label indicating the moment at which their receiver will be allowed to decrypt them.

**Definition 1.** A TRE scheme is a 5-uple of algorithms:

**TRE.Setup:** is a probabilistic algorithm run by a time server to generate system-wide parameters  $\mathbf{params}$  that include a public key  $\mathbf{TS}_{\text{pub}}$  for which the corresponding private key  $\mathbf{ts}_{\text{priv}}$  is stored in order to be used in all time-specific trapdoor generations.

**TRE.User-Keygen:** is a probabilistic algorithm taking as input public parameters  $\mathbf{params}$  that is run by each end-user to generate a key pair  $(\mathbf{upk}, \mathbf{usk})$ . The public keys are required to have a special form and their validity should be verifiable in polynomial time.

**TRE.TS-Release:** is an algorithm run by the server that, given  $\mathbf{ts}_{\text{priv}}$  and a time information  $T \in \{0, 1\}^t$ , generates and discloses a specific trapdoor  $s_T$ . The latter's validity should be verifiable in polynomial time given  $T \in \{0, 1\}^t$  and  $\mathbf{TS}_{\text{pub}}$ .

**TRE.Encrypt:** is a probabilistic algorithm taking as inputs public parameters  $\mathbf{params}$ , a recipient's public key  $\mathbf{upk}$ , a message  $m \in \mathcal{M}$  and a time information  $T \in \{0, 1\}^t$  to produce a ciphertext  $(C, T) = \text{TRE.Encrypt}(m, \mathbf{upk}, \mathbf{params}, T)$  that the recipient must be unable to decrypt before knowing  $s_T = \text{TRE.TS-release}(\mathbf{ts}_{\text{priv}}, T)$ .

**TRE.Decrypt:** is a deterministic algorithm taking as inputs a ciphertext  $(C, T)$ , parameters  $\mathbf{params}$ , a private key  $\mathbf{usk}$  and a time-specific trapdoor  $s_T$  to return a plaintext  $m$  or a distinguished symbol  $\perp$  if the ciphertext is not properly formed.

For consistency, we impose that  $\text{TRE.Decrypt}(\mathbf{usk}, s_T, \mathbf{params}, (C, T)) = m$  whenever  $(C, T) = \text{TRE.Encrypt}(m, \mathbf{upk}, \mathbf{params}, T) = m$  for all messages  $m \in \mathcal{M}$  if  $s_T = \text{TRE.TS-release}(\mathbf{ts}_{\text{priv}}, T)$ .

We distinguish two kinds of adversaries. We first consider malicious receivers attempting to gain information on the plaintext before its release-time. Such adversaries do not know the server's private key but can freely choose the public key on which they are challenged in a find-then-guess game. In both stages, they have access to a release-time oracle returning trapdoors for any arbitrary time periods but the (adversarially-chosen) one for which the challenge ciphertext is computed. In a chosen-ciphertext scenario, they are also given access to an oracle decrypting other ciphertexts than the challenge. These adversaries are called chosen-time period and ciphertext attackers (CTCA) in contrast to weaker chosen-time period and plaintext attackers (CTPA).

**Definition 2.** A TRE scheme is secure against chosen-time period and ciphertext attacks (IND-CTCA) if no probabilistic polynomial time (PPT) attacker has a non-negligible advantage in the following game:

1. Given a security parameter  $1^k$ , the challenger runs  $\text{TRE.Setup}(1^k)$  and gives the resulting parameters  $\mathbf{params}$  (that include the server's public key  $\mathbf{TS}_{\text{pub}}$ ) to  $\mathcal{A}$  while  $\mathbf{ts}_{\text{priv}}$  is kept secret.
2.  $\mathcal{A}$  queries a release-time oracle  $\text{TRE.TS-release}(\cdot)$  returning trapdoors  $s_T$  for arbitrary time periods  $T$  as well as a decryption oracle  $\text{TRE.Decrypt}(\cdot)$ .

which, given a ciphertext  $(C, T)$  and a receiver's public key  $\mathbf{upk}$  provided by  $\mathcal{A}$ , generates the decryption of  $C$  using the trapdoor  $s_T$  even without knowing the private key  $\mathbf{usk}$  corresponding to  $\mathbf{upk}$ . At some moment,  $\mathcal{A}$  outputs messages  $m_0, m_1$ , an arbitrary public key  $\mathbf{upk}^*$  and a time-period  $T^*$  that was not submitted to the  $TRE.TS\text{-release}(\cdot)$  oracle. She gets the challenge  $(C^*, T^*) = TRE.Encrypt(m_b, \mathbf{upk}^*, \mathit{params}, T^*)$ , for a hidden bit  $b \stackrel{R}{\leftarrow} \{0, 1\}$ .

3.  $\mathcal{A}$  issues new release-time queries for any arbitrary time-period but  $T^*$  and decryption queries for any ciphertext but the challenge  $(C^*, T^*)$  for the public key  $\mathbf{upk}^*$ . She eventually outputs a bit  $b'$  and wins if  $b' = b$ . As usual, her advantage is  $Adv_{TRE-IND-CCA}^{ind-cca}(\mathcal{A}) := 2 \times Pr[b' = b] - 1$ .

The above model of security against receivers is seemingly stronger than its counterpart in [28] for which target time periods are fixed by the challenger at the beginning of the game instead of being adaptively chosen by adversaries. When compared to the notion of 'recipient security' defined in [1] or its counterpart in [24], definition 2 also looks stronger as the authors of [1,24] explicitly omitted to provide the attacker with a decryption oracle and argued that such an oracle is useless since the receiver's private key is known to the adversary. Actually, she might still gain useful information by asking for the decryption of ciphertexts  $(C, T^*) \neq (C^*, T^*)$  for the target time period  $T^*$ . That is why, although the challenger does not a priori know any private key except the server's one, we provide the attacker with an oracle that is more powerful than an usual decryption oracle: given a time-information string, a receiver's public key  $\mathbf{upk}$  and a ciphertext, it either returns a plaintext or a rejection message even if it does not know the matching private key  $\mathbf{usk}$  for  $\mathbf{upk}$ .

The latter requirement might look too strong in practice but it is to be noted that a similar constraint was imposed by Al-Riyami and Paterson in their security model for certificateless encryption schemes (CLE) ([2,3]). As they did in their context, we can argue here that an adversary has more power if she can obtain the decryption of ciphertexts for receivers's public keys that she simply observes without knowing the matching private key. Besides, since the scheme that we propose in section 4.1 perfectly supports this constraint, we do not believe the latter to be too strong.

Finally, in the model of [1], the challenge key pair  $(\mathbf{usk}^*, \mathbf{upk}^*)$  is chosen by the challenger at the outset of the game. Our model does not assume  $\mathbf{usk}^*$  to be known to the challenger. It is unclear whether this distinction is of any practical relevance but it seems more natural to allow adversaries to be challenged on any receiver's public key of their choosing without directly revealing the associated private key (which is not needed to compute the challenge ciphertext after all). In fact, the knowledge of  $\mathbf{usk}^*$  is not needed in the security proof of our scheme.

In a second definition, we consider the threat of curious servers where attackers know the server's private key but are challenged on a random user's public key for which they are equipped with a decryption oracle.

**Definition 3.** A  $TRE$  scheme is said to be secure against chosen-ciphertext attacks (or  $IND\text{-}CCA$  secure) if no  $PPT$  adversary  $\mathcal{A}$  has a non-negligible advantage in the following game:

1. Given  $1^k$ , the challenger  $\mathcal{CH}$  runs the algorithms  $TRE.Setup(1^k)$  and  $TRE.User-Keygen$  to obtain a list of public parameters  $\mathbf{params}$  and a pair  $(\mathbf{upk}, \mathbf{usk})$ .  $\mathcal{CH}$  gives  $\mathbf{params}$ , the server's private key  $\mathbf{ts}_{priv}$  and the public key  $\mathbf{upk}$  to  $\mathcal{A}$  while the private key  $\mathbf{usk}$  is kept secret.
2.  $\mathcal{A}$  is given access to a decryption oracle  $TRE.Decrypt(\cdot)$  which, given a ciphertext  $(C, T)$  and the time-specific trapdoor  $\mathbf{s}_T$  (which is always computable for the adversary who knows  $\mathbf{ts}_{priv}$ ), returns the decryption of  $C$  using the private key  $\mathbf{usk}$ . At some point, she outputs equal-length messages  $m_0, m_1$  and a challenge time-period  $T^*$ . She gets a ciphertext  $(C^*, T^*) = TRE.Encrypt(m_b, \mathbf{upk}, \mathbf{params}, T^*)$ , for  $b \stackrel{R}{\leftarrow} \{0, 1\}$ , computed under the public key  $\mathbf{upk}$ .
3.  $\mathcal{A}$  issues a new sequence of queries but is prohibited from asking for the decryption of the challenge for the time period  $T^*$ . She eventually outputs a bit  $b'$  and wins if  $b' = b$ . Her advantage is still defined as  $Adv_{TR-CKE}^{ind-cca}(\mathcal{A}) := 2 \times Pr[b' = b] - 1$ .

In the full version of this paper, we establish the security of the Blake-Chan ([8]) scheme in our enhanced security model. The IND-CTCA security is proved under a stronger assumption than its counterpart in a weaker sense in [1].

### 3 Why CLE Does Not Imply TRE

The model of security formalized in definition 2 is reminiscent of the definition of security against Type I adversaries against certificateless encryption scheme (CLE) in that the challenger might have to answer decryption queries on ciphertexts presumably created using a public key for which it does not even know the private key. Besides, the scheme that we describe in section 5.2 bears similarities with a CLE scheme recently proposed in [3] in the same way as the Blake-Chan scheme ([8]) has salient similarities with the CLE scheme described in [2].

Actually, it turns out that some constructions may provide instantiations of both primitives but it is very unlikely that a generic transformation can turn a secure CLE into a secure TRE because of differences between formal models: in CLE schemes, some principal's public key is associated to any identity even though no explicit certificate is used. In contrast, time information strings are never bound to any public key.

It is very tempting to believe that a TRE scheme can be generically obtained from a CLE system by turning the Key Generation Center (KGC) into a time server and transforming the partial key private extraction algorithm (see [2] or [3] for details on certificateless primitives) into a release-time algorithm.

The problems arise when attempting to establish the security of the obtained scheme in the sense of definition 3 assuming that the underlying CLE is secure against malicious KGCs (called Type II adversaries in [2]). In the model of security against a Type II adversary ([2]), the latter is disallowed to replace public keys. Now, in the game of definition 3, consider what happens when the attacker issues a decryption query  $(C, T)$  for a completely arbitrary time period  $T$ . In the game that it plays against its own challenger, the challenger of definition 3 is

stuck as it may not replace the public key assigned to the entity of identity  $T$  with the challenge public key  $\text{upk}$  since replacement queries are forbidden.

Even worse: when the adversary of definition 3 produces her challenge request  $(m_0, m_1, T^*)$ , it is very likely that the challenge public key  $\text{upk}$  is not associated to  $T^*$  in the game played by the challenger against its own "certificateless challenger". It comes that, even in the chosen-plaintext scenario, the security of the underlying CLE scheme does not imply the security of the obtained TRE.

On the other hand, if the adversary was challenged on a fixed random user's key pair  $(\text{usk}^*, \text{upk}^*)$  provided by the challenger in the game of definition 2 as in the definition of 'receiver security' given in [1], the techniques of Dodis and Katz ([18]) would certainly yield a secure TRE by suitably combining an identity based encryption scheme (IBE) with a traditional public key encryption scheme. Nevertheless, because of the special decryption oracles used in definition 2 where the challenger does not even know adversarially controlled private keys, it is unclear whether the same techniques also apply here.

## 4 An Efficient TRE Construction Using Bilinear Maps

This section presents a new efficient timed-release encryption scheme. It makes use of *bilinear map groups* which are groups  $(\mathbb{G}_1, \mathbb{G}_2)$  of prime order  $p$  for which there exists a bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  satisfying the following properties:

1. Bilinearity:  $\forall u, v \in \mathbb{G}_1, \forall a, b \in \mathbb{Z}_p^*$ , we have  $e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degeneracy: if  $g$  generates  $\mathbb{G}_1$ , then  $e(g, g)$  generates  $\mathbb{G}_2$
3. Computability:  $\forall u, v \in \mathbb{G}_1, e(u, v)$  can be efficiently computed

The security of our construction is proved to rely on the intractability of the following problem that was introduced in [10].

The *q-Bilinear Diffie-Hellman Inversion Problem* (*q*-BDHIP) consists in, given  $(g, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^q}) \in \mathbb{G}_1^{q+1}$ , computing  $e(g, g)^{1/\alpha} \in \mathbb{G}_2$ .

### 4.1 The Scheme

In the new scheme, called **TRE1**, time-specific trapdoors are signatures computed using a signature scheme independently considered in [9] and [30] unlike the scheme of [8] that uses trapdoors computed according to Boneh et al.'s short signature algorithm ([13]). The **TRE1** scheme has similarities with a selective-ID secure IBE that was proven secure without random oracles in ([10]) but its security proofs hold in the random oracle model ([7]). The consistency of the scheme is easy to check as

$$e(X^{rh_1(T)} Y^r, \mathbf{s}_T)^{1/a} = e(g^{\alpha(s+h_1(T))}, g^{\frac{1}{\alpha(s+h_1(T))}})^r = e(g, g)^r.$$

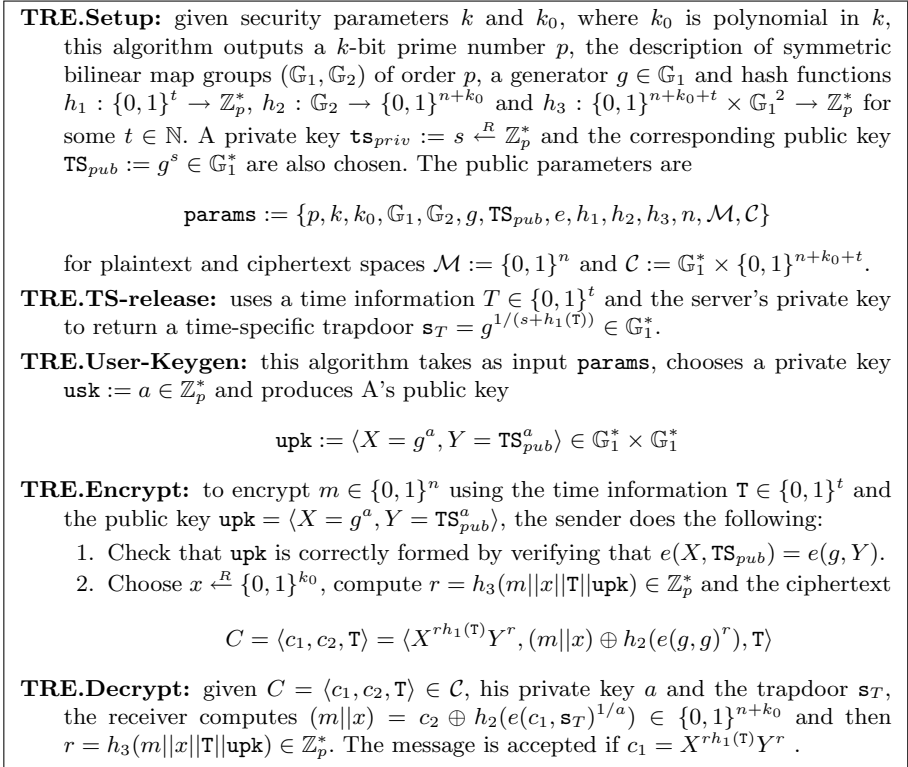


Fig. 1. The TRE1 scheme

### 4.2 Efficiency Discussions

If  $e(g, g)$  is included among the public parameters, an exponentiation in  $\mathbb{G}_2$  and a multi-exponentiation in  $\mathbb{G}_1$  are needed for both the sender and the receiver while the latter must also compute a pairing.

TRE1 is thus significantly more efficient than the scheme recently proposed by Blake and Chan ([8]) as no pairing must be computed at encryption. It actually happens to be more practical to encrypt messages with distinct release-times in succession for the same recipient. Indeed, the TRE scheme of [8] requires the sender to compute a pairing that depends on the release-time and, if Alice has to send several ciphertexts with distinct release-times to Bob, she has to compute a new pairing for each encryption. Moreover, TRE1 does not need a special (and much less efficient) hash function mapping strings onto a cyclic group (and it thus benefits from a faster release-time algorithm) while both schemes have similar complexities at decryption.

As for the scheme proposed in [8], the sender has to verify the validity of the public key (in step 1 of the encryption algorithm) to ensure that the recipient will be enabled to decrypt the message. Such a checking is fortunately needed only once at the first use of the key.

### 4.3 Security

We stress on the importance of including the public key  $\mathbf{upk}$  among the inputs of the hash function  $h_3$  because the scheme would be insecure in the game of definition 2 otherwise (as the adversary could turn the challenge into another encryption of the same plaintext for a different public key). In a security analysis, theorems 1 and 2 show that **TRE1** is secure in the sense of definitions 2 and 3. The proofs are detailed in the full version of the paper.

**Theorem 1.** *Assume that an IND-CTCA attacker  $\mathcal{A}$  has an advantage  $\epsilon$  against **TRE1** when running a time  $\tau$ , making  $q_{h_i}$  queries to random oracles  $h_i$  ( $i = 1, 2, 3$ ) and  $q_D$  decryption queries. Then the  $q$ -BDHIP can be solved for  $q = q_{h_1}$  with a probability*

$$\epsilon' > \frac{1}{(q_{h_1} + q_{h_3})(q_{h_2} + q_{h_3})} (\epsilon - q_{h_3}/2^{k_0-1})(1 - 2^{-k})^{q_D}$$

within a time  $\tau' < \tau + O((q_{h_1}^2 + q_{h_3})\tau_{exp})$  where  $\tau_{exp}$  is the maximum of the costs of an exponentiation in  $\mathbb{G}_1$  and in  $\mathbb{G}_2$ .

**Theorem 2.** *Assume that an IND-CCA attacker  $\mathcal{A}$  has an advantage  $\epsilon$  against **TRE1** when running a time  $\tau$ , making  $q_{h_i}$  queries to random oracles  $h_i$  ( $i = 1, 2, 3$ ) and  $q_D$  decryption queries. Then the 1-BDHIP can be solved with a probability  $\epsilon' > (q_{h_2} + q_{h_3})^{-1}(\epsilon - q_{h_3}/2^{k_0-1})(1 - 2^{-k})^{q_D}$  within a time  $\tau' < \tau + O(q_{h_3}\tau_{exp})$  where  $\tau_{exp}$  is the maximum time to perform an exponentiation in  $\mathbb{G}_1$  and in  $\mathbb{G}_2$ .*

As **TRE1** results from a variant of the first Fujisaki-Okamoto ([20]) transform applied to a simpler version of the scheme (details are given in the full paper), the proofs apply a variant of theorem 3 in ([20]).

### 4.4 Encrypting for Multiple Receivers

Interestingly, the scheme is practical to encrypt messages intended to several recipients *with the same release-time* (encrypting with distinct release-times is forbidden as colluding receivers could decrypt the message without having the appropriate trapdoor): given a plaintext  $m$  and public keys  $\mathbf{upk}_1 = (X_1, Y_1), \dots, \mathbf{upk}_N = (X_N, Y_N)$ , ciphertexts have the form

$$\langle X_1^{r h_1(\mathcal{T})} Y_1^r, \dots, X_N^{r h_1(\mathcal{T})} Y_N^r, (m||x) \oplus h_2(e(g, g)^r), \mathcal{L} \rangle$$

where  $r = h_3(m||x||\mathcal{T}||\mathbf{upk}_1||\dots||\mathbf{upk}_N)$  and  $\mathcal{L}$  is a label indicating how each part of ciphertext is associated to each receiver.

The sender still has no pairing to compute: only a multi-exponentiation per receiver (in addition to an exponentiation in  $\mathbb{G}_2$ ) is needed. The Blake-Chan scheme and its generalization ([1]) do not enjoy this efficiency as one pairing per receiver must be computed.



The security proofs are straightforward adaptations of the proofs of theorems 1 and 2 in a security model which is a simple extension of the one described in section 3: in the extension of definition 2, the adversary outputs a set of  $N$  public keys at the end of the find stage whereas, in the counterpart of definition 3, she is challenged a vector of  $N$  public keys.

## 5 Adding Release-Time Confidentiality

In the security model considered by Di Crescenzo et al. ([17]), the time server is required to interact with the receiver so that the latter obtains the message if the current time exceeds the release-time but nothing can be learned about the latter by the server.

However, as release-times appended to ciphertexts are transmitted in clear to receivers in their model as in ours, nothing can prevent a spying server (or anyone else) observing release-times of ciphertexts from attempting to gain information on who their recipient could be upon release of the corresponding trapdoor by watching who enquires about it within a reasonably small set of users. Such a threat would hamper the key privacy property ([5]) that TRE1 could be shown to satisfy in an adapted security model if release-times were scrambled.

We believe that, in order to minimize the server's knowledge about who is talking to whom and enhance the protocol's anonymity, it may be desirable to even preclude such a scenario and guarantee the confidentiality of release-times against anyone but intended recipients who can first unmask a part of the received ciphertext using their private key and learn the release-time before obtaining the corresponding trapdoor. We thus define a new notion called *release-time confidentiality* that captures the inability for the server to decide under which out of two release-times of its choice a given ciphertext was created.

**Definition 4.** A TRE scheme is said to provide **release-time confidentiality** (or IND-RT-CCA security) if no PPT adversary  $\mathcal{A}$  has a non-negligible advantage in the game below:

1. Given  $\mathbf{1}^k$ , the challenger  $\mathcal{CH}$  runs the algorithms  $TRE.Setup(\mathbf{1}^k)$  and  $TRE.User-Keygen$  to obtain a list of public parameters  $\mathbf{params}$  and a pair  $(\mathbf{upk}, \mathbf{usk})$ .  $\mathcal{CH}$  gives  $\mathbf{params}$ , the server's private key  $\mathbf{ts}_{priv}$  and the public key  $\mathbf{upk}$  to  $\mathcal{A}$  while the private key  $\mathbf{usk}$  is kept secret.
2.  $\mathcal{A}$  is given access to a decryption oracle  $TRE.Decrypt(\cdot)$  which, given a ciphertext  $(C, T)$  and the time-specific trapdoor  $\mathbf{s}_T$  (which is always computable for the adversary who knows  $\mathbf{ts}_{priv}$ ), returns the decryption of  $C$  using the private key  $\mathbf{usk}$ . At some moment, she outputs a plaintext  $m^*$  and two time periods  $T_0^*, T_1^*$  before getting a challenge  $C^* = TRE.Encrypt(m^*, \mathbf{upk}, \mathbf{params}, T_b^*)$ , for  $b \xleftarrow{\mathcal{R}} \{0, 1\}$ .
3.  $\mathcal{A}$  issues a new sequence of queries but she is of course prohibited from requesting the decryption of  $C^*$  under the time periods  $T_b^*$ . She eventually outputs a bit  $b'$  and wins if  $b' = b$ . Her advantage is  $Adv_{TR-PKE}^{ind-rt-cca}(\mathcal{A}) := 2 \times Pr[b' = b] - 1$ .

### 5.1 The TRE1 Case

The TRE1 construction does not provide the confidentiality of release-times as they must be appended to ciphertexts and thus transmitted in clear. However, for applications that would require it, a very simple modification of TRE1 satisfies the new property at the cost of a slight increase in the workload of the sender who has to compute an additional multi-exponentiation while the complexity of the decryption algorithm remains unchanged. The only change is that, instead of being transmitted in clear within the ciphertext, the release-time  $T$  is scrambled using a hash value of  $c'_1 = g^{r^{h_1(T)}TS_{pub}^r}$  (obtained from an additional random oracle  $h_4$ ) which is also  $c'_1^{1/a}$  so that the receiver can first unmask it before obtaining the trapdoor.

In the random oracle model, the modified scheme, called TRE2, has the release-time confidentiality property under the standard Diffie-Hellman assumption in  $\mathbb{G}_1$  (in order for this new security notion to rely on the latter assumption, we need to feed hash function  $h_2$  with both  $c'_1$  and  $e(g, g)^r$  in the encryption algorithm) as claimed by theorem 3.

**Theorem 3.** *Assume that an attacker  $\mathcal{A}$  has an advantage  $\epsilon$  against the release time confidentiality of TRE2 in the sense of definition 4 when running a time  $\tau$ , making  $q_{h_i}$  queries to random oracles  $h_i$  ( $i = 1, \dots, 4$ ) and  $q_D$  decryption queries. Then there is an algorithm  $\mathcal{B}$  solving the computational Diffie-Hellman problem with a probability*

$$\epsilon' > (\epsilon - q_{h_3}/2^{k_0-1})(1 - 2^{-k})^{q_D}$$

*within a time  $\tau' < \tau + O(q_{h_3}\tau_{exp}) + O((2q_{h_3} + q_{h_2} + q_{h_4})\tau_p)$  where  $\tau_{exp}$  is the maximum time to perform an exponentiation in  $\mathbb{G}_1$  and in  $\mathbb{G}_2$  and  $\tau_p$  is the cost of a pairing computation.*

### 5.2 Release-Time Confidentiality in the Blake-Chan TRE

A very simple method allows adding the release time confidentiality property to the scheme proposed in [8] at a minimal cost: a single additional exponentiation in  $\mathbb{G}_1$  is required at encryption while the decryption operation has essentially the same cost as in the original scheme.

Interestingly, this modification allows proving the security under a weaker assumption than for the original version (details will be given in the full version of the paper): the IND-CTPA security is showed under the bilinear Diffie-Hellman assumption while the IND-CPA and IND-RT-CPA securities both rely on the hardness of the standard Diffie-Hellman problem. As for TRE1 and TRE2, the chosen-ciphertext security is obtained via similar transformations to [20,21].

## 6 Conclusion

We proposed a new stringent security model for non-interactive timed-release encryption schemes and presented a new efficient construction fitting this model.

**TRE.Setup:** given a security parameters  $k$ , this algorithm chooses a  $k$ -bit prime number  $p$ , symmetric bilinear map groups  $(\mathbb{G}_1, \mathbb{G}_2)$  of order  $p$ , a generator  $g \in \mathbb{G}_1$  and hash functions  $h_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ ,  $h_2 : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \{0, 1\}^n$  and  $h_3 : \mathbb{G}_1 \rightarrow \{0, 1\}^t$ . It also selects a private key  $\mathbf{ts}_{priv} := s \xleftarrow{R} \mathbb{Z}_p^*$  and sets  $\mathbf{TS}_{pub} := g^s \in \mathbb{G}_1^*$  as the corresponding public key. The ciphertext space is  $\mathcal{C} := \mathbb{G}_1^* \times \{0, 1\}^{n+t}$  while the space of plaintexts is  $\mathcal{M} := \{0, 1\}^n$ . The public parameters are then

$$\mathbf{params} := \{k, p, \mathbb{G}_1, \mathbb{G}_2, g, \mathbf{TS}_{pub}, e, h_1, h_2, n, \mathcal{M}, \mathcal{C}\}.$$

**TRE.TS-release:** given  $\mathbf{T} \in \{0, 1\}^t$ , the server discloses a trapdoor  $\mathbf{s}_T = h_1(\mathbf{T})^s$ .

**TRE.User-Keygen:** this algorithm takes as input  $\mathbf{params}$ , chooses a private key  $\mathbf{usk} := a \in \mathbb{Z}_p^*$  and produces A's public key  $\mathbf{upk} := X = g^a \in \mathbb{G}_1^*$ .

**TRE.Encrypt:** to encrypt  $m \in \{0, 1\}^n$  for the time period  $T \in \{0, 1\}^t$  and the public key  $\mathbf{upk} = X = g^a$ , the sender chooses  $r \xleftarrow{R} \mathbb{Z}_p^*$  and the ciphertext is

$$C = \langle c_1, c_2, c_3 \rangle = \langle g^r, m \oplus h_2(X^r || e(\mathbf{TS}_{pub}, h_1(\mathbf{T}))^r), \mathbf{T} \oplus h_3(X^r) \rangle$$

**TRE.Decrypt:** given a ciphertext  $C = \langle c_1, c_2, c_3 \rangle \in \mathcal{C}$ , a private key  $a \in \mathbb{Z}_p^*$ , the receiver computes  $c'_1 = c_1^a \in \mathbb{G}_1^*$  to obtain  $\mathbf{T} = c_3 \oplus h_3(c'_1) \in \{0, 1\}^t$  and recover the plaintext  $m = c_2 \oplus h_2(c'_1 || e(c_1, \mathbf{s}_T)) \in \{0, 1\}^n$  upon release of  $\mathbf{s}_T$ .

**Fig. 2.** The BC-TRE2 scheme

We also explained how to enhance the anonymity of ciphertexts at a minimum cost in our scheme as in Blake and Chan's one in accordance with a new formally defined security property.

## References

1. S.-S. Al-Riyami , J. Malone-Lee, N.P. Smart, *Escrow-Free Encryption Supporting Cryptographic Workflow*, available from <http://eprint.iacr.org/2004/258>.
2. S.-S. Al-Riyami , K.G. Paterson, *Certificateless Public Key Cryptography*, in *Advances in Cryptology - Asiacypt'03*, LNCS 2894, pp. 452–473, Springer, 2003.
3. S.S. Al-Riyami , K.G. Paterson, *CBE from CL-PKE: A Generic Construction and Efficient Schemes* , in *proc. of PKC'05*, LNCS 3386, pp. 398–415, Springer, 2005.
4. J. Baek, R. Safavi-Naini, W. Susilo, *Token-Controlled Public Key Encryption*, to appear in *proc. of ISPEC'05*, LNCS series, 2005.
5. M. Bellare, A. Boldyreva, A. Desai, D. Pointcheval, *Key-Privacy in Public-Key Encryption*, in *Advances in Cryptology - Asiacypt'01*, LNCS 2248, pp. 566–582. Springer, 2001.
6. M. Bellare, S. Goldwasser, *Encapsulated key-escrow*, 4<sup>th</sup> ACM Conference on Computer and Communications Security, 1997.
7. M. Bellare, P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, 1<sup>st</sup> ACM Conference on Computer and Communications Security, pp. 62-73, 1993.
8. I. Blake, A.-C.-F. Chan, *Scalable, Server-Passive, User-Anonymous Timed Release Public Key Encryption from Bilinear Pairing*, available from <http://eprint.iacr.org/2004/211/>, 2004.

9. D. Boneh, X. Boyen, *Short Signatures Without Random Oracles*, in Advances in Cryptology - Eurocrypt'04, LNCS 3027, Springer, pp. 56–73, 2004.
10. D. Boneh, X. Boyen, *Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles*, in Advances in Cryptology - Eurocrypt'04, LNCS 3027, Springer, pp. 223–238, 2004.
11. D. Boneh, X. Boyen, E.-J. Goh, *Hierarchical Identity Based Encryption with Constant Size Ciphertext*, available at <http://eprint.iacr.org/2005/015>.
12. D. Boneh, M. Franklin, *Identity Based Encryption From the Weil Pairing*, in Advances in Cryptology - Crypto'01, LNCS 2139, pp. 213–229, Springer, 2001.
13. D. Boneh, B. Lynn, H. Shacham, *Short signatures from the Weil pairing*, in Advances in Cryptology - Asiacrypt'01, LNCS 2248, pp. 514–532, Springer, 2001.
14. D. Boneh, M. Naor, *Timed Commitments*, Advances in Cryptology - Crypto'00, LNCS 1880, pp. 236–254, Springer, 2000.
15. R. Canetti, S. Halevi, J. Katz, *A Forward Secure Public Key Encryption Scheme*, Advances in Cryptology - Eurocrypt'03, LNCS 2656, pp. 254–271, Springer, 2003.
16. L. Chen, K. Harrison, N. Smart, D. Soldera, *Applications of Multiple Trust Authorities in Pairing Based Cryptosystems*, in Infracsec'02, LNCS 2437, pp. 260–275, Springer, 2002.
17. G. Di Crescenzo, R. Ostrovsky, S. Rajagopalan, *Conditional Oblivious Transfer and Timed-Release Encryption*, in Advances in Cryptology - Eurocrypt'99, LNCS 1592, pp. 74–89, Springer, 1999.
18. Y. Dodis, J. Katz, *Chosen-Ciphertext Security of Multiple Encryption*, in TCC'05, LNCS 3378, pp. 188–209, Springer, 2005.
19. Y. Dodis, D.-H. Yum, *Time Capsule Signatures*, to appear in proc. of Financial Cryptography 2005, LNCS series, 2005.
20. E. Fujisaki, T. Okamoto, *How to Enhance the Security of Public-Key Encryption at Minimum Cost*, in proc. of PKC'99, LNCS 1560, pp. 53–68. Springer, 1999.
21. E. Fujisaki and T. Okamoto, *Secure integration of asymmetric and symmetric encryption schemes*, in Advances in Cryptology - Crypto'99, LNCS 1666, pp. 537–554. Springer, 1999.
22. J. Garay, M. Jakobsson, *Timed-Release of Standard Digital Signatures*, in Financial Crypto'02, LNCS 2357, pp. 168–182, Springer, 2002.
23. J. Garay, C. Pomerance, *Timed Fair Exchange of Standard Signatures*, in Financial Crypto'03, LNCS 2742, pp. 190–207, Springer, 2003.
24. Y. H. Hwang, D. H. Yum, P. J. Lee *Timed-Release Encryption with Pre-open Capability and its Application to Certified E-mail System*, to appear in ISC'05, LNCS series, 2005.
25. W. Mao, *Timed-Release Cryptography*, in Selected Areas in Cryptography'01, LNCS 2259, pp. 342–357, Springer, 2001.
26. T. May, *Time-release crypto*, manuscript, February 1993.
27. M.C. Mont, K. Harrison. M. Sadler, *The HP time vault service: Innovating the way confidential information is disclosed at the right time*, in 12th International World Wide Web Conference, pp. 160–169, ACM Press, 2003.
28. I. Osipkov, Y. Kim, J.-H. Cheon, *Timed-Release Public Key Based Authenticated Encryption*, available from <http://eprint.iacr.org/2004/231>.
29. R. Rivest, A. Shamir, D.A. Wagner, *Time-lock puzzles and timed-release crypto*, MIT LCS Tech. Report MIT/LCS/TR-684, 1996.
30. F. Zhang, R. Safavi-Naini, W. Susilo, *An Efficient Signature Scheme from Bilinear Pairings and Its Applications*, in proc. of PKC'04, LNCS 2947, pp. 277–290, 2004.

# Security Properties of Two Authenticated Conference Key Agreement Protocols

Qiang Tang and Chris J. Mitchell

Information Security Group, Royal Holloway, University of London,  
Egham, Surrey TW20 0EX, UK  
{qiang.tang, c.mitchell}@rhul.ac.uk

**Abstract.** In this paper we analyse the security of two authenticated group key agreement schemes based on the group key agreement protocol of Burmester and Desmedt. One scheme was proposed by Burmester and Desmedt, and uses a separate authentication scheme to achieve authentication among the participants. We show that this scheme suffers from a number of security vulnerabilities. The other scheme was generated using the general protocol compiler of Katz and Yung. We show that in some circumstances, even if key confirmation is implemented, this scheme still suffers from insider attacks (which are not covered by the security model used by Katz and Yung).

## 1 Introduction

Since the pioneering work of Diffie and Hellman [1], key agreement has become a very active and fruitful research area in cryptography. The case of two-party key agreement has been well investigated, and a variety of provably secure schemes have been proposed (e.g., [2,3,4]). However, less attention has been given to group key agreement, which enables more than two participants to negotiate a session key. Of especial interest are authenticated group key agreement protocols, designed for use in a hostile environment where communications are over open networks which may be fully controlled by an adversary.

Of the existing group key agreement protocols, a number are based on the idea of extending the two-party Diffie-Hellman protocol [1] to the group setting (e.g., [5,6,7,8,9,10]). Among these schemes, the cyclic group key agreement protocol due to Burmester and Desmedt (here referred to as the BD scheme) is particularly efficient; it has been rigorously proved to be secure against a passive adversary [11]. A number of authenticated group key agreement schemes based on the BD scheme have been proposed, including those in [5,12,13,14,15,16]. In this paper, we focus on the enhanced BD scheme [5] and a scheme due to Katz and Yung [15], referred to below as the KY scheme.

In the enhanced BD scheme, an interactive zero-knowledge proof scheme is used to achieve authentication among the conference participants, while in the KY scheme a signature mechanism is used to achieve authentication among the conference participants. Both schemes are more efficient than the scheme

of Bresson et al. [17], which was the first authenticated group key agreement scheme proved to be secure in a formal model.

The main contribution of this paper lies in analysing the security properties of the BD scheme, and exhibiting potential security vulnerabilities in the enhanced BD scheme and the KY scheme. The rest of this paper is organised as follows. In section 2, we review three group key agreement schemes, namely the BD scheme, the enhanced BD scheme, and the KY scheme. In section 3, we give our observations on the security of these three schemes. In section 4, we conclude this paper.

## 2 Review of the Target Schemes

In all three schemes, the following parameters are made public during the initialisation stage:  $G$  is a multiplicative group with large prime order  $q$ , and  $g$  is a generator of  $G$ . We suppose all the potential participants and their identities are from the set  $\{(U_1, ID_{U_1}), \dots, (U_m, ID_{U_m})\}$ , where  $m$  is a sufficiently large integer and  $ID_{U_i}$  ( $1 \leq i \leq n \leq m$ ) is the identity of  $U_i$ .

### 2.1 Description of the BD Scheme

Suppose a set  $S = \{U_1, \dots, U_n\}$  ( $n \leq m$ ) of users wish to establish a session key; then each user  $U_i$  ( $1 \leq i \leq n$ ) performs the following steps. It should be noted that the indices of users (and values exchanged between users) are taken modulo  $n$ .

1.  $U_i$  chooses a random  $s_i$  ( $0 \leq s_i < q$ ), and broadcasts  $Z_i = g^{s_i}$ .
2. After receiving  $Z_{i-1}$  and  $Z_{i+1}$ ,  $U_i$  computes and broadcasts  $X_i$ :

$$X_i = (Z_{i+1}/Z_{i-1})^{s_i}$$

3. After receiving every  $X_j$  ( $1 \leq j \leq n$ ),  $U_i$  computes the session key  $K_i$  as:

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{ns_{i-1}s_i} \cdot \left(\frac{g^{s_i s_{i+1}}}{g^{s_{i-1}s_i}}\right)^{n-1} \cdot \left(\frac{g^{s_{i+1}s_{i+2}}}{g^{s_i s_{i+1}}}\right)^{n-2} \cdots \frac{g^{s_{i+n-2}s_{i+n-1}}}{g^{s_{i+n-3}s_{i+n-2}}} \\ &= g^{s_{i-1}s_i + s_i s_{i+1} + s_{i+1}s_{i+2} + \cdots + s_{i+n-2}s_{i+n-1}} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1} \end{aligned}$$

If all the participants are honest, then all of them will compute the same session key because  $K_1 = \dots = K_n$ . In [11], Burmester and Desmedt prove that this scheme is secure against a passive adversary.

### 2.2 Description of the Enhanced BD Scheme

The enhanced BD scheme provides partial authentication for the protocol messages by using an authentication scheme which is secure against adaptive chosen text attacks. The authentication scheme and the enhanced BD scheme operate as follows.

**The authentication scheme.** In the initialisation stage, the system selects four large primes  $p_2, p_3, q_2, q_3$  satisfying  $p_2 \leq q_3$ ,  $q_2|(p_2 - 1)$ , and  $q_3|(p_3 - 1)$ . Let  $g_2$  be a generator of a multiplicative group of order  $q_2$  in  $Z_{p_2}^*$ , and  $g_3$  be a generator of a multiplicative group of order  $q_3$  in  $Z_{p_3}^*$ . Each user  $U_i$  ( $1 \leq i \leq n$ ) in the system publishes his public key  $\{\beta_{i2}, \beta_{i3}, \gamma_{i3}\}$ , where  $\{\beta_{i2} = g_2^{a_{i2}}, \beta_{i3} = g_3^{a_{i3}}, \gamma_{i3} = g_3^{b_{i3}}\}$ , and keeps  $\{a_{i2}, a_{i3}, b_{i3}\}$  secret as his private key.

Suppose  $U_i$  wishes to prove knowledge of  $z$  to  $U_j$  ( $j \neq i$ ); the authentication scheme operates as follows.  $U_i$  sends  $z$  and  $\gamma_{i2} = g_2^{b_{i2}z}$  to  $U_j$ , where  $b_{i2}$  is randomly selected ( $0 \leq b_{i2} \leq q_2$ ). Simultaneously  $U_i$  proves to  $U_j$  that he knows the discrete logarithm base  $g_2$  of  $\beta_{i2}^z \gamma_{i2}$  and the discrete logarithm base  $g_3$  of  $\beta_{i3}^{\gamma_{i2}} \gamma_{i3}$ , using the zero-knowledge discrete logarithm proof scheme of Chaum et al. [18], described below.  $U_j$  checks that  $\gamma_{i2}^{q_2} \equiv 1 \pmod{p_2}$ ,  $g_2^{q_2} \equiv \beta_{i2}^{q_2} \equiv 1 \pmod{p_2}$ ,  $g_3^{q_3} \equiv \beta_{i3}^{q_3} \equiv \gamma_{i3}^{q_3} \pmod{p_3}$ , that  $q_2, q_3$  are primes, and that  $p_2 \leq q_3$ . If any of the checks fail,  $U_j$  terminates the protocol. Otherwise  $U_j$  now believes that  $U_i$  knows  $z$ .

The Chaum et al. zero knowledge discrete logarithm proof scheme [18] operates as follows. Suppose  $P$  is a large prime, and that  $\alpha^x \equiv \beta \pmod{P}$ . Suppose also that  $P, \alpha, \beta$  are made public and  $x$  is a secret of Alice. If Alice wants to prove her knowledge of  $x$  to Bob, she performs the following steps.

1. Alice selects  $T$  random numbers  $e_i$  ( $0 \leq e_i < P - 1, 1 \leq i \leq T$ ). Alice computes and sends  $h_i = \alpha^{e_i} \pmod{P}$  ( $1 \leq i \leq T$ ) to Bob.
2. Bob chooses and sends  $T$  random bits  $b_i \in \{0, 1\}$  ( $1 \leq i \leq T$ ) to Alice.
3. For each bit  $b_i$  ( $1 \leq i \leq T$ ), if  $b_i = 0$  Alice sets  $s_i = e_i$ ; otherwise Alice computes  $s_i = e_i - e_j \pmod{P - 1}$ , where  $j$  is the minimal number for which  $b_j = 1$ . Finally, Alice sends  $(x - e_j) \pmod{P - 1}$  and  $s_i$  ( $1 \leq i \leq T$ ) to Bob.
4. For each bit  $i$  ( $1 \leq i \leq T$ ), if  $b_i = 0$  Bob checks  $\alpha^{s_i} = h_i$ ; otherwise Bob checks that  $\alpha^{s_i} = h_i h_j^{-1}$ . Then Bob checks  $\alpha^{x - e_j} = \beta h_j^{-1}$ .

If all the checks succeed, Bob can confirm with a probability of  $1 - (\frac{1}{2})^T$  that Alice knows  $x$  [18].

Burmester and Desmedt [5] claim that the above scheme is a secure authentication system, i.e., it has the following three security properties:

1. When only a passive adversary is present,  $U_i$  can successfully prove his knowledge of  $z$  to  $U_j$  with an overwhelming probability.
2. If an attacker impersonates  $U_i$ , then  $U_j$  can detect it with an overwhelming probability.
3. If an active attacker substitutes  $z$  with  $z'$  ( $z \neq z'$ ), then  $U_j$  will reject it with an overwhelming probability.

**The enhanced BD scheme.** Suppose a set  $S = \{U_1, \dots, U_n\}$  ( $n \leq m$ ) of users wish to establish a session key; then each user  $U_i$  ( $1 \leq i \leq n$ ) performs the following steps. Note that the indices of users (and values exchanged between users) are taken modulo  $n$ .

1.  $U_i$  chooses a random  $s_i$  ( $0 \leq s_i \leq q$ ), and computes and broadcasts  $Z_i = g^{s_i}$ .
2. After receiving  $Z_{i-1}$  and  $Z_{i+1}$ ,  $U_i$  proves his knowledge of  $s_i$  to  $U_{i+1}$ , and verifies  $U_{i-1}$ 's knowledge of  $s_{i-1}$ .

If both the proof and the verification succeed,  $U_i$  computes and broadcasts  $X_i$ :

$$X_i = (Z_{i+1}/Z_{i-1})^{s_i}$$

3. After receiving  $X_j$  ( $1 \leq j \leq n$ ),  $U_i$  computes the session key  $K_i$ :

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{ns_{i-1}s_i} \cdot \left(\frac{g^{s_i s_{i+1}}}{g^{s_{i-1}s_i}}\right)^{n-1} \cdot \left(\frac{g^{s_{i+1}s_{i+2}}}{g^{s_i s_{i+1}}}\right)^{n-2} \cdots \frac{g^{s_{i+n-2}s_{i+n-1}}}{g^{s_{i+n-3}s_{i+n-2}}} \\ &= g^{s_{i-1}s_i + s_i s_{i+1} + s_{i+1}s_{i+2} + \cdots + s_{i+n-2}s_{i+n-1}} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1} \end{aligned}$$

If all the participants are honest, then all of them will compute the same session key because  $K_1 = \cdots = K_n$ .

Burmester and Desmedt [5] claim that the enhanced BD scheme is a secure key agreement scheme, i.e., in a protocol instance it is computationally infeasible for any set of active attackers to compute the same session key as that which is computed by the honest participants.

### 2.3 Description of the KY Scheme

Katz and Yung [15] proposed a general protocol compiler that can transform a group key agreement protocol secure against a passive adversary into an authenticated group key agreement protocol secure against both passive and active adversaries. As an example, Katz and Yung transformed the unauthenticated BD scheme into an authenticated group key agreement protocol. Katz and Yung [15] prove that this protocol is secure against an active adversary, i.e., the advantage of any probabilistic polynomial time (PPT) active adversary is negligible.

The KY scheme [15] requires that, during the initialisation stage, each user  $U_i$  ( $1 \leq i \leq m$ ) generates a verification/signing key pair  $(PK_{U_i}, SK_{U_i})$  by running  $Gen(1^k)^1$ , where  $k$  is a security parameter. Suppose a set  $S = \{U_1, \dots, U_n\}$  ( $n \leq m$ ) of users wish to establish a session key; then each user  $U_i$  ( $1 \leq i \leq n$ ) performs the following steps. It should be noted that, as previously, the indices of users (and values exchanged between users) are taken modulo  $n$ . Throughout this paper,  $\parallel$  represents the string concatenation operator.

1.  $U_i$  chooses a random  $r_i$  ( $0 \leq r_i < q$ ) and broadcasts  $ID_{U_i}$ , 0, and  $r_i$ .
2. After receiving the broadcast messages from all other participants,  $U_i$  sets  $nonce_i = ((ID_{U_1}, r_1), \dots, (ID_{U_n}, r_n))$  and stores it as part of its state information<sup>2</sup>.

<sup>1</sup> We suppose that  $\Sigma = (Gen, Sign, Vrfy)$  is a signature scheme which is strongly unforgeable under adaptive chosen message attack (as defined in [15]).

<sup>2</sup> If all the messages are transported correctly, every user will possess the same state information.



$U_i$  chooses a random number  $s_i$  ( $0 \leq s_i < q$ ) and computes  $Z_i = g^{s_i}$ . Then  $U_i$  computes the signature  $\sigma_{i1} = \text{Sign}_{SK_{U_i}}(1||Z_i||\text{nonce}_i)$  and broadcasts  $ID_{U_i}, 1, Z_i,$  and  $\sigma_{i1}$ .

3. When  $U_i$  receives the message  $ID_{U_j}, 1, Z_j,$  and  $\sigma_{j1}$  from user  $U_j$  ( $1 \leq j \leq n, j \neq i$ ), he checks that: (1)  $U_j$  is an intended participant, (2) 1 is the next expected sequence number for a message from  $U_j$ , and (3)  $\sigma_{j1}$  is a valid signature, i.e.  $\text{Vrfy}_{PK_{U_j}}(1||Z_j||\text{nonce}_i, \sigma_{j1}) = 1$ , where an output of 1 signifies acceptance. If any of these checks fail,  $U_i$  terminates the protocol. Otherwise,  $U_i$  computes  $X_i = (Z_{i+1}/Z_{i-1})^{s_i}$  and the signature  $\sigma_{i2} = \text{Sign}_{SK_{U_i}}(2||X_i||\text{nonce}_i)$ . Then  $U_i$  broadcasts  $ID_{U_i}, 2, X_i,$  and  $\sigma_{i2}$ .
4. When  $U_i$  receives the message  $ID_{U_j}, 2, X_j,$  and  $\sigma_{j2}$  from user  $U_j$  ( $1 \leq j \leq n, j \neq i$ ), he checks that: (1)  $U_j$  is an intended participant, (2) 2 is the next expected sequence number for a message from  $U_j$ , and (3)  $\sigma_{j2}$  is a valid signature, i.e.  $\text{Vrfy}_{PK_{U_j}}(2||X_j||\text{nonce}_i, \sigma_{j2}) = 1$ . If any of these checks fail,  $U_i$  terminates the protocol. Then  $U_i$  computes the session key  $K_i$ :

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{ns_{i-1}s_i} \cdot \left(\frac{g^{s_i s_{i+1}}}{g^{s_{i-1}s_i}}\right)^{n-1} \cdot \left(\frac{g^{s_{i+1}s_{i+2}}}{g^{s_i s_{i+1}}}\right)^{n-2} \cdots \frac{g^{s_{i+n-2}s_{i+n-1}}}{g^{s_{i+n-3}s_{i+n-2}}} \\ &= g^{s_{i-1}s_i + s_i s_{i+1} + s_{i+1}s_{i+2} + \cdots + s_{i+n-2}s_{i+n-1}} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1} \end{aligned}$$

If all the participants are honest, then all of them will compute the same session key  $K = K_1 = \cdots = K_n = g^{s_1 s_2 + s_2 s_3 + \cdots + s_n s_1}$ .

Katz and Yung [15] also proposed the following method (without security proof) to achieve key confirmation for the authenticated group key agreement scheme: after computing key  $K$ , each player  $U_i$  computes  $x_i = F_K(ID_{U_i})$ , signs  $x_i$ , and broadcasts  $x_i$  and the corresponding signature, where  $F$  represents a pseudo-random function. However, they did not specify how the signature is computed. In this paper, we suppose the signature is computed as  $\sigma_{ij} = \text{Sign}_{SK_{U_i}}(x_i)$ , where  $j$  is the next round number.

However, it should be noted that a different key confirmation method is proposed in the full version of [15], and this method is proved to be secure in a different security model in [19].

### 3 Properties of the Target Schemes

In this section we first describe certain security properties of the BD scheme, and then demonstrate security vulnerabilities in both the enhanced BD scheme and the KY scheme.

#### 3.1 Security Properties of the BD Scheme

In the BD scheme, a malicious participant, say  $U_j$  ( $1 \leq j \leq n$ ), who can manipulate the communications in the network, is able to make any other participant,

say  $U_i$  ( $1 \leq i \leq n$ ,  $i \neq j$ ), compute the session key to be any value  $K^* \in G$  chosen by  $U_j$ .

To achieve this, in the second step  $U_j$  intercepts the message  $X_{i-n+2}$  and prevents it from reaching  $U_i$ .  $U_j$  then waits until all the other messages have been received and computes the session key  $K$  in the normal way, i.e. as in step 3 of section 2.1.  $U_j$  now sends  $X'_{i+n-2} = X_{i+n-2} \cdot \frac{K^*}{K}$  to  $U_i$ , pretending that it comes from  $U_{i+n-2}$ .

**Lemma 1.** *As a result of the above attack,  $U_i$  will compute the session key as  $K^*$ .*

*Proof.* This is immediate, since  $U_i$  will compute the session key as  $K \cdot \frac{X'_{i+n-2}}{X_{i+n-2}} = K^*$ , by definition of  $X'_{i+n-2}$ .

In summary, in the BD scheme any participant capable of manipulating the messages received by another participant can completely control the value of the session key obtained by that participant. In the following subsections, we show that this property means that schemes derived from the BD scheme possess certain security vulnerabilities.

### 3.2 Security Vulnerabilities in the Enhanced BD Scheme

The enhanced BD scheme suffers from the following potential security vulnerabilities.

**Man-in-the-middle attack.** To mount a man-in-the-middle attack, an active adversary proceeds as follows. In the first step of the protocol the adversary replaces the message  $Z_{i+1}$  sent to  $U_i$  with  $Z'_{i+1} = Z_{i-1}^2$ , for every  $i$  ( $1 \leq i \leq n$ ). Then we can prove that the protocol will end successfully and the adversary can compute the session key held by  $U_i$  ( $1 \leq i \leq n$ ).

**Lemma 2.** *Under the above attack, the protocol will end successfully, and the adversary can compute the session key held by  $U_i$  for every  $i$  ( $1 \leq i \leq n$ ).*

*Proof.* Under the attack, it is clear that the protocol will end successfully, because it is only required that  $U_i$  ( $1 \leq i \leq n$ ) proves his knowledge of  $s_i$  to  $U_{i+1}$  (while the adversary only changes the message that  $U_{i+1}$  sends to  $U_i$ ).

It is also clear that, in the second step,  $U_i$  will broadcast  $X_i = (Z'_{i+1}/Z_{i-1})^{s_i} = (Z_{i-1})^{s_i}$ . Then, after intercepting all the broadcast values  $X_i$  ( $1 \leq i \leq n$ ), the adversary can compute the session key held by  $U_i$  as

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= (X_i)^n \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \end{aligned}$$

which involves only values broadcast by the various recipients. The result follows.

**Insider different key attack.** In the enhanced BD scheme, it is only required that  $U_i$  ( $1 \leq i \leq n$ ) proves his knowledge of  $s_i$  to  $U_{i+1}$ . The authentication requirement can successfully prevent a malicious attacker from impersonating  $U_i$  ( $1 \leq i \leq n$ ) to send a forged message  $Z_i$  to the honest participant  $U_{i+1}$ . However, a malicious participant, say  $U_j$  ( $1 \leq j \leq n$ ), who can manipulate the communications in the network, is still able to make any other participant, say  $U_i$  ( $1 \leq i \leq n, i \neq j$ ), compute the session key to be any value  $K^* \in G$ .

In fact, any active outsider attacker can also mount such an attack by manipulating  $X_i$  ( $1 \leq i \leq n$ ) in step 2 of the protocol run, but the attacker cannot obtain any information about the session keys obtained by the legitimate participants.

**Outsider impersonation attack.** An outsider attacker can also impersonate a valid participant,  $U_i$  say, in some circumstances, but the attacker cannot obtain the session key. This attack is based on the following security vulnerability in the Chaum et al. zero-knowledge discrete logarithm proof scheme<sup>3</sup>. The vulnerability arises from the fact that proof scheme does not enable the prover to specify the verifier.

Suppose Alice wishes to prove her knowledge of  $x$  to Bob, then an attacker can concurrently impersonate Alice to prove knowledge of  $x$  to any other entity, Carol say. The attack can be mounted as follows.

1. Alice selects  $T$  random numbers  $e_i$  ( $0 \leq e_i \leq P-1, 1 \leq i \leq T$ ), and computes and sends  $h_i = \alpha^{e_i} \bmod P$  ( $1 \leq i \leq T$ ) to Bob.  
The attacker intercepts the message, prevents it from reaching Bob, and forwards  $h_i = \alpha^{e_i} \bmod P$  ( $1 \leq i \leq T$ ) to Carol pretending to be Alice (i.e. starting a second run of the protocol).
2. Carol chooses and sends random bits  $e_i \in \{0, 1\}$  ( $1 \leq i \leq T$ ) to the attacker as part of the second protocol run. The attacker then impersonates Bob to forward  $e_i \in \{0, 1\}$  ( $1 \leq i \leq T$ ) to Alice as the second message of the first protocol run.
3. For each bit  $b_i$  ( $1 \leq i \leq T$ ), if  $b_i = 0$  Alice sets  $s_i = e_i$ ; otherwise Alice computes  $s_i = e_i - e_j \bmod P - 1$ , where  $j$  is the minimal number that  $b_j = 1$ . Alice sends  $x - e_j$  and  $s_i$  ( $1 \leq i \leq T$ ) to Bob as the third message of the first protocol run. The attacker intercepts the message, prevents it from reaching Bob, and forwards it to Carol as the third message of the second protocol run.
4. For each bit  $i$  ( $1 \leq i \leq T$ ), if  $b_i = 0$  Carol checks  $\alpha^{s_i} = h_i$ ; otherwise Carol checks that  $\alpha^{s_i} = h_i h_j^{-1}$ . Then Carol checks that  $\alpha^{x - e_j} = \beta h_j^{-1}$ .

It is easy to verify that Carol's checks will succeed and confirm that the attacker knows  $x$ . This attack conflicts with the claim made in [5] that the authentication technique is secure against any type of attack.

<sup>3</sup> It should be noted that this vulnerability exists not only in this specific scheme, but also exists in all such schemes with only a one-way proof of knowledge.

We now show how to use the above observation to attack the enhanced BD scheme. Suppose the attacker detects that a set  $S = \{U_1, \dots, U_n\}$  of users is starting the key agreement protocol to negotiate a session key (referred to below as the first protocol instance). The attacker impersonates  $U_i$  to initiate a second instance of the key agreement protocol among a different set  $S' = \{U'_1, U'_2, \dots, U'_{n'}\}$  of users, where  $U'_i = U_i$ . In these two protocol instances, the attacker performs the following steps.

1. In the first protocol instance,  $U_i$  chooses a random  $s_i$  ( $0 \leq s_i < q$ ), and computes and broadcasts  $Z_i = g^{s_i}$ . The attacker intercepts the messages from both  $U_{i-1}$  and  $U_{i+1}$  to  $U_i$  and prevents them from reaching  $U_i$ . In the second protocol instance, the attacker impersonates  $U_i$  to broadcast  $Z_i = g^{s_i}$ . Other participants perform as required by the protocol. Suppose the messages sent by  $U'_{i-1}$  and  $U'_{i+1}$  are  $Z'_{i-1}$  and  $Z'_{i+1}$ . The attacker impersonates  $U_{i-1}$  and  $U_{i+1}$  to send  $Z'_{i-1}$  and  $Z'_{i+1}$  to  $U_i$  in the first protocol instance.
2. In the first protocol instance, when  $U_i$  proves his knowledge of  $s_i$  to  $U_{i+1}$ , the attacker mounts the above attack against the Chaum et al. zero-knowledge discrete logarithm proof scheme by impersonating  $U_i$  to prove his knowledge of  $s_i$  to  $U'_{i+1}$  in the second protocol instance. In the second protocol instance, when  $U'_{i-1}$  proves his knowledge of  $s'_{i-1}$  to the attacker, the attacker mounts the above attack against the Chaum et al. zero-knowledge discrete logarithm proof scheme by impersonating  $U'_{i-1}$  to prove  $U'_{i-1}$ 's knowledge of  $s'_{i-1}$  to  $U_i$  in the first protocol instance. In the first protocol instance  $U_i$  computes and broadcasts  $X_i$  as:

$$X_i = (Z'_{i+1}/Z'_{i-1})^{s_i}$$

The attacker intercepts this message and impersonates  $U_i$  to broadcast the same message in the second protocol instance.

3. In the second protocol instance, the users  $U'_j$  ( $1 \leq j \leq n', j \neq i$ ) computes the common session key. However, the attacker cannot compute the session key because he does not know  $s_i$ . The first instance will be terminated because the authentication messages between  $U_i$  and  $U_{i+1}$  are blocked by the attacker.

In the second protocol instance, the attacker succeeds in impersonating  $U_i$  to the members of the set  $S'$ .

**Insider impersonation attack.** In the above attack, suppose that the attacker is a legitimate participant in the second protocol instance, i.e. suppose that the attacker is  $U'_j$  in the set  $S'$  ( $U'_j$  is not required to be a member of the set  $S$ ). Then  $U'_j$  can successfully impersonate  $U_i$  in the second protocol instance without any knowledge of the secret key of  $U_i$ . In this case, it is clear that  $U'_j$  can compute the session key agreed by all the participants of  $S'$ , since  $U'_j$  is a legitimate member of  $S'$ .

This attack means that, even if authentication is implemented, a malicious participant can still impersonate another honest participant in a protocol instance.

### 3.3 Potential Security Vulnerability in the KY Scheme

We show that the KY scheme suffers from insider different key attacks even if key confirmation is implemented. Specifically, we show that any  $n - 2$  malicious participants can make the other honest participants compute different keys at the end of the protocol. However, it should be noted that insider attacks are not covered by the security model in [15]. Recently, insider attacks have been modelled by Katz and Shin in [19].

For simplicity we describe the attack in three-party case. Suppose, in some past successful instance of the KY protocol among  $\{U_1, U_2, U_3\}$ , the key confirmation message sent by  $U_3$  is  $x_3^* = F_{K_3^*}(ID_{U_3})$  and  $\sigma_{33}^* = \text{Sign}_{SK_{U_3}}(x_3^*)$ .  $U_2$  can initiate a new instance of the KY protocol among  $\{U_1, U_2, U_3\}$  and mount a different key attack as follows.

1. Each user  $U_i$  ( $1 \leq i \leq 3$ ) begins by choosing a random  $r_i$  ( $0 \leq r_i < q$ ) and broadcasting  $ID_{U_i}$ , 0, and  $r_i$ .
2. After receiving the initial broadcast messages,  $U_i$  ( $1 \leq i \leq 3$ ) sets  $\text{nonce}_i = ((ID_{U_1}, r_1), (ID_{U_2}, r_2), (ID_{U_3}, r_3))$  and stores it as part of its state information. Then  $U_i$  chooses a random  $s_i$  ( $0 \leq s_i < q$ ), computes  $Z_i = g^{s_i}$  and the signature  $\sigma_{i1} = \text{Sign}_{SK_{U_i}}(1||Z_i||\text{nonce}_i)$ , and then broadcasts  $ID_{U_i}$ , 1,  $Z_i$ , and  $\sigma_{i1}$ .
3. When  $U_i$  ( $1 \leq i \leq 3$ ) receives the messages from other participants, he checks the messages as required by the protocol. It is easy to verify that all the checks will succeed.

$U_1$  computes and then broadcasts  $ID_{U_1}$ , 2,  $X_1$ , and  $\sigma_{12}$ , where

$$X_1 = (Z_2/Z_3)^{s_1}, \sigma_{12} = \text{Sign}_{SK_{U_1}}(2||X_1||\text{nonce}_1).$$

$U_3$  computes and then broadcasts  $ID_{U_3}$ , 2,  $X_3$ , and  $\sigma_{32}$ , where

$$X_3 = (Z_1/Z_2)^{r_3}, \sigma_{32} = \text{Sign}_{SK_{U_3}}(2||X_3||\text{nonce}_3)$$

$U_2$  computes and sends  $ID_{U_2}$ , 2,  $X_2$ , and  $\sigma_{22}$  to  $U_3$ , where

$$X_2 = (Z_3/Z_1)^{s_2}, \sigma_{22} = \text{Sign}_{SK_{U_2}}(2||X_2||\text{nonce}_2)$$

$U_2$  then waits until all the other messages have been received and computes the session key  $K$  in the normal way, i.e. as in step 3 of section 2.1.  $U_2$  now sends  $ID_{U_2}$ , 2,  $X'_2$ , and  $\sigma'_{22}$  to  $U_1$ , where

$$X'_2 = X_2 \cdot \frac{K_3^*}{K}, \sigma'_{22} = \text{Sign}_{SK_{U_2}}(2||X'_2||\text{nonce}_2)$$

**Lemma 3.** *As a result of the above steps,  $U_1$  will compute the session key as  $K_3^*$ , and  $U_3$  will compute the session key as  $K$ .*

*Proof.* This is immediate, since  $U_1$  will compute the session key as  $K_3^* = (Z_3)^{3s_1}(X_1)^2X'_2 = K \cdot \frac{K_3^*}{K}$ , and  $U_3$  will compute the session key as  $K = (Z_2)^{3s_3}(X_3)^2X_1$ .

Hence, as a result,  $U_2$  shares the session keys  $K_3^*$  and  $K$  ( $K \neq K_3^*$ ) with  $U_1$  and  $U_3$  respectively.

4.  $U_2$  intercepts the confirmation messages between  $U_1$  and  $U_3$  and prevents them from reaching their indeed destinations.  $U_2$  computes and sends the confirmation messages  $x'_2 = F_{K_3^*}(ID_{U_2})$  and  $\sigma'_{23} = \text{Sign}_{SK_{U_2}}(x'_2)$  to  $U_1$ , and then sends  $x_2 = F_K(ID_{U_2})$  and  $\sigma_{23} = \text{Sign}_{SK_{U_2}}(x_2)$  to  $U_3$ . Then  $U_2$  impersonates  $U_3$  to send  $x_{33}^*$  and  $\sigma_{33}^*$  to  $U_1$ .

$U_2$  initiates a second instance of the key agreement protocol among the members of a set  $S''$ , which includes  $U_1$  and  $U_2$ . In step 3 of the new instance,  $U_2$  manipulates the communications and forces  $U_1$  to compute the session key as  $K$ , and then obtains the confirmation message ( $x_1 = F_K(ID_{U_1})$ ,  $\sigma_{13} = \text{Sign}_{SK_{U_1}}(x_1)$ ) from  $U_1$ .

$U_2$  impersonates  $U_1$  to forward ( $x_1 = F_K(ID_{U_1})$ ,  $\sigma_{13} = \text{Sign}_{SK_{U_1}}(x_1)$ ) to  $U_3$  in the current protocol instance.

5. It is easy to verify that all the key confirmation messages will be checked successfully by the various participants, and the attack will therefore succeed.

It is clear that this security vulnerability can be removed if every user  $U_i$  ( $1 \leq i \leq 3$ ) is required to compute his key confirmation message as  $x_i = F_K(ID_{U_i})$ ,  $\sigma_{i3} = \text{Sign}_{SK_{U_i}}(3||x_i||\text{nonce}_i)$ .

## 4 Conclusions

In this paper we have shown that a number of security vulnerabilities exist in both the enhanced BD scheme and the KY scheme. In particular, we have shown that in the enhanced BD scheme the implementation of the authentication scheme does not meet the authentication requirement specified in [5]. One possible way of removing the vulnerabilities in the enhanced BD scheme would be to authenticate each message using a digital signature, as in the KY scheme.

## Acknowledgements

The authors would like to express their deep appreciation for the valuable comments provided by Jonathan Katz and Kenneth G. Paterson.

## References

1. Diffie, W., Hellman, M.: New directions in cryptography. *IEEE Transactions on Information Theory* **IT-22** (1976) 644–654
2. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In Stinson, D.R., ed.: *Advances in Cryptology – Crypto '93*. Volume 773 of *Lecture Notes in Computer Science.*, Springer-Verlag (1993) 110–125
3. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In Pfitzmann, B., ed.: *Advances in Cryptology — Eurocrypt 2001*. Volume 2045 of *Lecture Notes in Computer Science.*, Springer-Verlag (2001) 453–474

4. Chen, L., Kudla, C.: Identity based authenticated key agreement protocols from pairings. In: Proc. of the 16th IEEE Computer Security Foundations Workshop — CSFW 2003, IEEE Computer Society Press (2003) 219–233
5. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system. In Santis, A.D., ed.: *Advances in Cryptology—EUROCRYPT '94*. Volume 950 of *Lecture Notes in Computer Science.*, Springer-Verlag (1994) 275–286
6. Ingemarsson, I., Tang, D., Wong, C.: A conference key distribution system. *IEEE Transactions on Information Theory* **28** (1982) 714–720
7. Kim, Y., Perrig, A., Tsudik, G.: Communication-efficient group key agreement. In: Proc. IFIP TC11 16th Annual Working Conference on Information Security. (2001) 229–244
8. Steer, D., Strawczynski, L., Diffie, W., Wiener, M.: A secure audio teleconference system. In Krawczyk, H., ed.: *Advances in Cryptology — Crypto '98*. Volume 403 of *Lecture Notes in Computer Science.*, Springer-Verlag (1998) 520–528
9. Tzeng, W.: A practical and secure-fault-tolerant conference-key agreement protocol. In Imai, H., Zheng, Y., eds.: *Proceedings of Public Key Cryptography: Third International Workshop on Practice and Theory in Public Key Cryptosystems*, Springer-Verlag (2000) 1–13
10. Tzeng, W.: A secure fault-tolerant conference-key agreement protocol. *IEEE Transactions on Computers* **51** (2002) 373–379
11. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system. In Santis, A.D., ed.: *Pre-Proceedings of Eurocrypt '94*. (1994) 279–290
12. Choi, K.Y., Hwang, J.Y., Lee, D.H.: Efficient ID-based group key agreement with bilinear maps. In Bao, F., Deng, R., Zhou, J.Y., eds.: *Proceedings of the 2004 International Workshop on Practice and Theory in Public Key Cryptography (PKC '04)*. Volume 2947 of *Lecture Notes in Computer Science.*, Springer-Verlag (2004) 130–144
13. Du, X.J., Wang, Y., Ge, J.H., Wang, Y.M.: ID-based authenticated two round multiparty key agreement. *Cryptology ePrint Archive: Report 2003/247* (2003)
14. Du, X.J., Wang, Y., Ge, J.H., Wang, Y.M.: An improved ID-based authenticated group key agreement scheme. *Cryptology ePrint Archive, Report 2003/260* (2003)
15. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In Boneh, D., ed.: *Advances in Cryptology — Crypto 2003*. Volume 2729 of *Lecture Notes in Computer Science.*, Springer-Verlag (2003) 110–125
16. Zhang, F.G., Chen, X.F.: Attacks on two ID-based authenticated group key agreement schemes. *Cryptology ePrint Archive, Report 2003/259* (2003)
17. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably authenticated group Diffie-Hellman key exchange. In: *Proceedings of the 8th ACM Conference on Computer and Communications Security*, ACM Press (2001) 255–264
18. Chaum, D., Evertse, J.H., Graaf, J., Peralta, R.: Demonstrating possession of a discrete logarithm without revealing it. In Odlyzko, A.M., ed.: *Advances in Cryptology—CRYPTO '86*, Springer-Verlag (1987) 200–212
19. Katz, J., Shin, J.: Modeling insider attacks on group key-exchange protocols. *Cryptology ePrint Archive: Report 2005/163* (2005)

# Cryptanalysis of Two User Identification Schemes with Key Distribution Preserving Anonymity

Eun-Jun Yoon and Kee-Young Yoo\*

Department of Computer Engineering, Kyungpook National University,  
Daegu 702-701, South Korea  
Tel.: +82-53-950-5553; Fax: +82-53-957-4846  
ejyoon@infosec.knu.ac.kr, yook@knu.ac.kr

**Abstract.** In 2004, Wu-Hsu proposed an efficient identification scheme preserving anonymity. However, Yang et al. showed that Wu-Hsu's scheme has a serious weakness, by which the service provider can learn the secret token of the user who requests services. To overcome this limitation, they further proposed a scheme to attain the same set of objectives as the previous works. Nevertheless, the two schemes still have other serious weaknesses. Accordingly, the current paper demonstrates the vulnerability of the two schemes. Furthermore, we present a method to avoid attack.

**Keywords:** Cryptography, Password, Key establishment, Forward Secrecy.

## 1 Introduction

In distributed computing environments, it is necessary to maintain user anonymity. That is, only the service provider can identify the user, while no other entity can determine any information concerning the user's identity. In 2000, Lee and Chang [1] proposed a user identification scheme based on the security of the factoring problem [2][3] and the one-way hash function [3][4]. Their scheme has the following advantages: (1) Users can request services without revealing their identities to the public; (2) Each user needs to maintain only one secret; (3) It is not necessary for service providers to record the password files for the users; (4) No master key updating is needed if a new service provider is added into the system.

However, in 2004, Wu-Hsu (WH) [5] showed that Lee-Chang's user identification scheme is insecure under two attacks. First, when a user requests service from a service provider, since only one-way authentication of the user is implemented, an attacker can impersonate the service provider; second, if an expired session key is disclosed, an attacker can break the user anonymity of the corresponding past session. Then they proposed a more efficient identification scheme

---

\* Corresponding author.



preserving the same merits [5]. The WH scheme not only effectively eliminates the security leaks of the Lee-Chang scheme, it also reduces computational complexities and communication costs.

Recently, Yang et al. (YWBWD) [6] showed that the WH scheme has a serious weakness, by which the service provider can learn the secret token of the user who requests services. To overcome this limitation, they further proposed a scheme to attain the same set of objectives as the previous works.

However, the WH scheme and YWBWD scheme have other serious weaknesses. Accordingly, the current paper demonstrates the vulnerability of two schemes. Using our attacks, we will show that a malicious user (including the service provider) can easily obtain a specific legal user's secret token and impersonate this specific user to request a service from the service provider and gain access privilege. Additionally, we will show that a malicious user (including the legal user) can easily get the service provider's secret token and impersonate this service provider to exchange a common session key with a legal user. Furthermore, we present an improvement to repair the security flaws of the two schemes.

This paper is organized as follows: In Section 2, we briefly review the WH scheme and YWBWD scheme. Section 3 shows the security flaws of two schemes. In Section 4, we present an improvement of the two schemes. In Section 5, we analyze the security of our proposed scheme. Finally, our conclusions are presented in Section 6.

## 2 Literature Review

This section separately reviews the WH scheme [5] and YWBWD scheme [6].

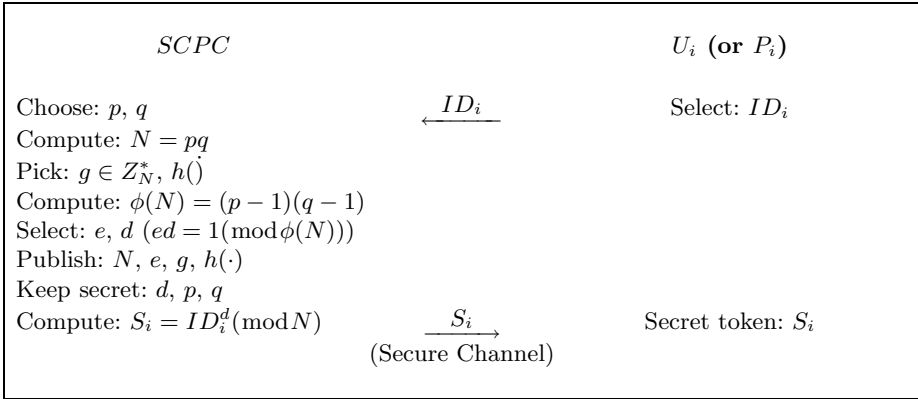
### 2.1 WH Scheme

The WH scheme is composed of two phases: key generation and anonymous user identification.

**Key Generation Phase:** The key generation phase of the WH scheme, which is illustrated in Figure 1, is as follows: A smart card producing center (*SCPC*) first chooses two large primes  $p$  and  $q$ , computes  $N = pq$ , picks an element  $g \in Z_N^*$  and a hash function  $h(\cdot)$ , and selects  $e$  and  $d$  such that  $ed = 1(\text{mod}\phi(N))$ , where  $\phi(N) = (p-1)(q-1)$  is the Euler totient function.  $N$ ,  $e$ ,  $g$  and  $h(\cdot)$  are published and  $d$ ,  $p$ , and  $q$  are kept secret by *SCPC*. Then, *SCPC* sends each user  $U_i$  (or service provider  $P_i$ ) a secret token  $S_i$  with a secure channel, where  $S_i = ID_i^d(\text{mod}N)$  and  $ID_i$  is the identity of  $U_i$  (or  $P_i$ ).

**Anonymous User Identification Phase:** The anonymous user identification phase of WH scheme, as illustrated in Figure 2, is as follows:

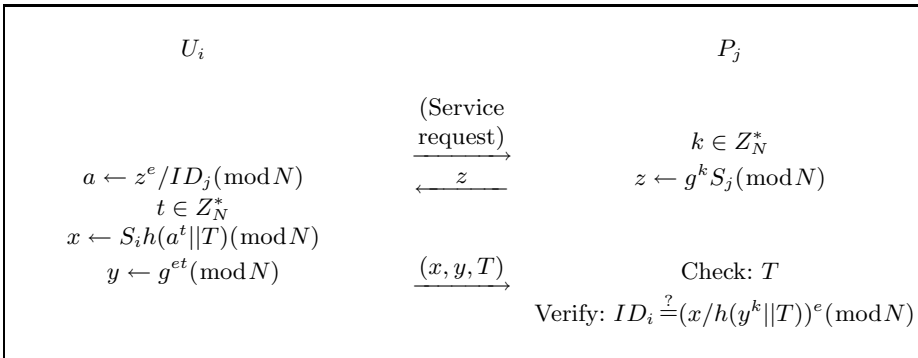
- (1)  $U_i$  first submits a service request to  $P_j$  to request a service from the service provider  $P_j$ .



**Fig. 1.** Key Generation Phase of WH scheme

- (2) After receiving the request,  $P_j$  chooses a random number  $k$  and computes  $z$ , where  $z = g^k S_j \pmod{N}$ . Then,  $P_j$  sends  $z$  to  $U_i$ .
- (3)  $U_i$  randomly chooses a number  $t$  and computes  $a, x$ , and  $y$ , where  $a = z^e / ID_j \pmod{N}$ ,  $x = S_i h(a^t || T) \pmod{N}$ ,  $y = g^{et} \pmod{N}$ , and  $T$  is the timestamp. Then,  $U_i$  sends  $(x, y, T)$  to  $P_j$ .
- (4) Finally,  $P_j$  checks  $T$  and verifies the equality  $ID_i \stackrel{?}{=} (x/h(y^k || T))^e \pmod{N}$ . If it holds for some  $ID_i$  existing in the identity list,  $U_i$  is accepted as an authorized user and the service request will be granted.

The user and the service provider share common session key as  $k_{ij} = a^{tx} = y^{kx} = g^{ektx} \pmod{N}$ , which can be used in subsequent communications for confidentiality.



**Fig. 2.** Anonymous User Identification Phase of WH scheme

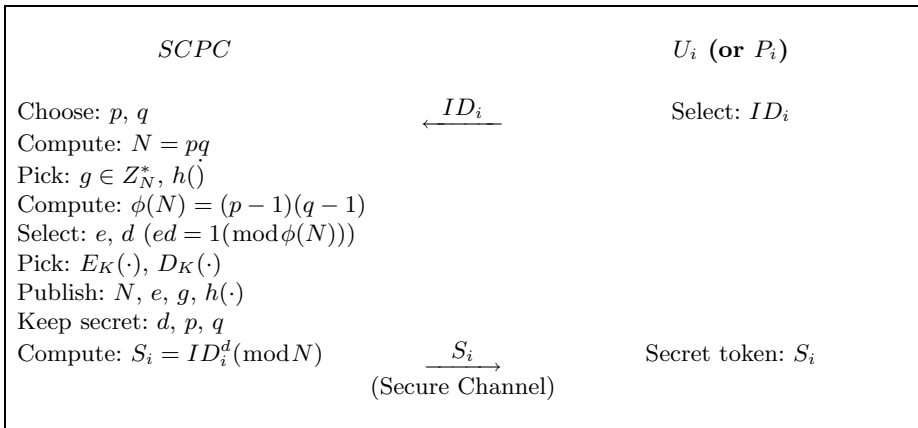
### 2.2 YWBWD Scheme

To solve the security problem in the WH scheme, Yang et al. proposed an improved version of the WH scheme. The YWBWD scheme is also composed of two phases; key generation and anonymous user identification.

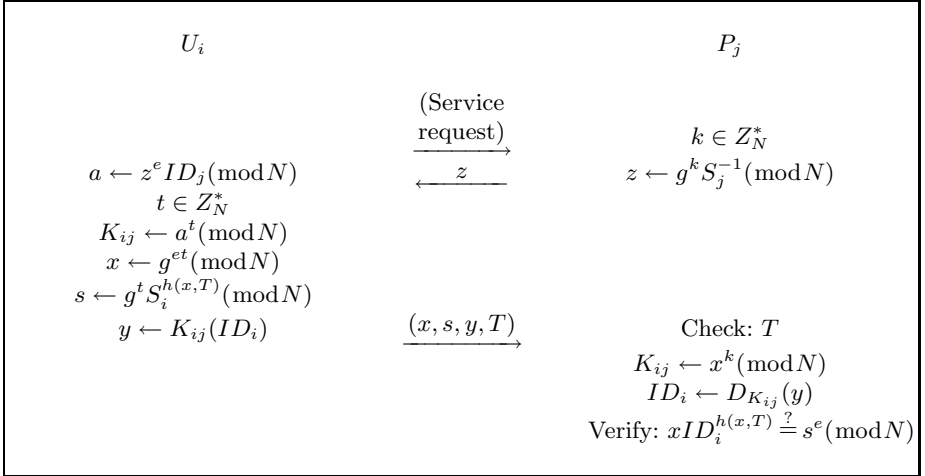
**Key Generation Phase:** The key generation phase in the YWBWD scheme is similar to that of the WH scheme. The key generation phase of YWBWD scheme, which is illustrated in Figure 3, is as follows: The smart card producing center (*SCPC*) first chooses two large primes  $p$  and  $q$ , computes  $N = pq$ , picks an element  $g \in Z_N^*$  (which is the generator of both  $Z_p$  and  $Z_q$ ) and a hash function  $h(\cdot)$ , and selects  $e$  and  $d$  such that  $ed = 1(\text{mod}\phi(N))$ , where  $\phi(N)(= (p - 1)(q - 1))$  is the Euler totient function. Note that  $e$  must be sufficiently large, e.g., 160 bits. Additionally, *SCPC* picks a symmetric-key cryptosystem such as DES Schneier, 1996, whose encryption function and decryption function under the private key  $K$  are  $E_K(\cdot)$  and  $D_K(\cdot)$ , respectively.  $N$ ,  $e$ ,  $g$ , and  $h(\cdot)$  are published and  $d$ ,  $p$ , and  $q$  are kept secret by *SCPC*. Then, *SCPC* sends each user  $U_i$  (or service provider  $P_i$ ) a secret token  $S_i$  with a secure channel, where  $S_i = ID_i^d(\text{mod}N)$  and  $ID_i$  is the identity of  $U_i$  (or  $P_i$ ).

**Anonymous User Identification Phase:** The anonymous user identification phase of the YWBWD scheme, which is illustrated in Figure 4, is as follows:

- (1)  $U_i$  first submits a service request to  $P_j$  to request a service from the service provider  $P_j$ .
- (2) After receiving the request,  $P_j$  chooses a random number  $k$  and computes  $z$ , where  $z = g^k S_j^{-1}(\text{mod}N)$ . Then,  $P_j$  sends  $z$  to  $U_i$ .
- (3)  $U_i$  randomly chooses a number  $t$  and computes  $a$ ,  $K_{ij}$ ,  $x$ ,  $s$ , and  $y$ , where  $a = z^e ID_j(\text{mod}N)$ ,  $K_{ij} = a^t(\text{mod}N)$ ,  $x = g^{et}(\text{mod}N)$ ,  $s = g^t S_i^{h(x,T)}(\text{mod}N)$ ,



**Fig. 3.** Key Generation Phase of YWBWD scheme



**Fig. 4.** Anonymous User Identification Phase of YWBWD scheme

$y = K_{ij}(ID_i)$ , and  $T$  is the timestamp. Then,  $U_i$  sends  $(x, s, y, T)$  to  $P_j$ . Note that  $K_{ij}$  is the common session key.

- (4) Finally,  $P_j$  first checks  $T$ . If it is old,  $P_j$  aborts the protocol. Otherwise,  $P_j$  obtains the common session key  $K_{ij} = x^k \pmod{N}$ . With  $K_{ij}$ ,  $P_j$  proceeds to decrypt  $y$  as  $ID_i = D_{K_{ij}}(y)$ .  $P_j$  then checks whether  $ID_i$  is on his maintained list. If  $ID_i$  is a legitimate user,  $P_j$  verifies the equality  $x ID_i^{h(x,T)} \stackrel{?}{=} s^e \pmod{N}$ . If the verification passes, then the service request is granted. Otherwise, the request is rejected.

The user and the service provider share common session key as  $k_{ij} = a^t = x^k = g^{ekt} \pmod{N}$ , which can be used in the subsequent communications for confidentiality.

### 3 Cryptanalysis of Two Schemes

This section show the security flaws of the WH scheme and YWBWD scheme. In the schemes, an attacker can freely impersonate the users or the service provider. This happens because an attacker can obtain the secret token  $S_i(S_j)$  of the user (or the service provider) after successful execution of the key generation phase.

#### 3.1 Attack to $U_i$

Suppose user  $U_f$  is an attacker who knows the legal user  $U_i$ 's  $ID_i$ . Usually, because the legal user's  $ID_i$  does not require safety, an attacker can easily get the target user's  $ID_i$  by various attack methods, such as stolen-verifier attacks [7] and server data eavesdropping [8]. For example, service provider  $P_j$  is always the target of attacker, because numerous users' secrets are stored in their databases.

The user  $ID$  table list stored in the service provider  $P_j$  can be eavesdropped and then used to impersonate the original user. By using the legal user  $U_i$ 's  $ID_i$ , in the key generation phase,  $U_f$  can register with  $SCPC$  as follows:

- (1)  $U_f$  obtains his/her identity  $ID_f$  by  $ID_f = ID_i^{-1}$  and submits  $ID_f$  as registration request to  $SCPC$ .
- (2)  $SCPC$  will compute the secret token  $S_f$  of  $U_f$  by  $S_f = ID_f^d = ID_i^{-d} = S_i^{-1}(\text{mod}N)$  and send  $S_f$  to  $U_f$  with a secure channel.

As a result,  $U_f$  can obtain the secret token  $S_i$  of the legal user  $U_i$  by computing  $S_f^{-1} = S_i(\text{mod}N)$ . Then, by using the  $S_i$ , so obtained,  $U_f$  can freely impersonate  $U_i$  to request a service from  $P_j$  and thus gain access privilege.

### 3.2 Attack to $P_j$

Suppose user  $U_f$  is an attacker who knows the the service provider  $P_j$ 's  $ID_j$ . In the key generation phase,  $U_f$  can register with  $SCPC$  as follows:

- (1)  $U_f$  obtains his/her identity  $ID_f$  by  $ID_f = ID_j^{-1}$  and submits  $ID_f$  as registration request to  $SCPC$ .
- (2)  $SCPC$  will compute the secret token  $S_f$  of  $U_f$  by  $S_f = ID_f^d = ID_j^{-d} = S_j^{-1}(\text{mod}N)$  and send  $S_f$  to  $U_f$  with a secure channel.

As a result,  $U_f$  can obtain the secret token  $S_j$  of the service provider  $P_j$  by computing  $S_f^{-1} = S_j(\text{mod}N)$ . Then, by using obtained  $S_j$ ,  $U_f$  can impersonate  $P_j$  and exchange a common session key with legal user  $U_i$ .

### 3.3 Another Attack

As another attack on two schemes, if a malicious  $U_i$  or  $P_j$ , who knows his/her  $S_i$  or  $S_j$ , computes his/her new identity  $ID_f$  by  $ID_f = ID_i ID_j$  and resubmits  $ID_f$  as a registration request to  $SCPC$ . Then,  $SCPC$  will compute the secret token  $S_f$  of  $U_f$  by  $S_f = ID_f^d = (ID_i ID_j)^d = S_j S_j(\text{mod}N)$  and send  $S_f$  to  $U_f$  with a secure channel. Consequently, a malicious  $U_i$  can obtain the secret token  $S_i$  of the legal user  $U_i$  or  $S_j$  of the service provider  $P_j$  by computing  $S_i = S_f S_j^{-1}(\text{mod}N)$  or  $S_j = S_f S_i^{-1}(\text{mod}N)$ , respectively.

## 4 Improved Scheme

This section presents a modification of the two schemes to correct the security flaws described in Section 3.

The proposed scheme employs the concept of hiding identity to prevent from above attacks. We only modify the key generation phase which issues a "hashed" identity for every legal user. That is, in the key generation phase, the smart card



## 6 Conclusions

The current paper demonstrated the security flaws of the WH user identification scheme and YWBWD user identification scheme. Using our attacks, we have shown that a malicious user (including a service provider) can easily get a specific legal user's secret token and impersonate this specific user to request a service from the service provider and gain access privilege. Additionally, we have shown that a malicious user (including the legal user) can easily get the service provider's secret token and impersonate this service provider to exchange a common session key with a legal user. For the above attacks, we presented an improvement to repair the security flaws of the two schemes.

## Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments in improving our manuscript. This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

## References

1. Lee, W.B., Chang, C.C.: User Identification and Key Distribution Maintaining Anonymity for Distributed Computer Network. *Computer Systems Science and Engineering*. Vol. 15. No. 4. (2000) 113-116
2. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signature and Public-key Cryptosystem. *Commun ACM*. Vol. 21. No. 2. (1978) 120-126
3. Schneier, B.: *Applied Cryptography*. 2nd ed. John Wiley & Sons. Inc. (1996)
4. Diffie, W., Hellman, M.: New Directions in Cryptography. *IEEE Trans Inf Theory*. Vol. 22. No. 6. (1976) 644-654
5. Wu, T.S., Hsu, C.L.: Efficient User Identification Scheme with Key Distribution Preserving Anonymity for Distributed Computer Networks. *Computer & Security*. Vol. 23. No. 2. (2004) 120-125
6. Yang, Y.J., Wang, S.H., Bao, F., Wang, J., Deng, R.H.: New Efficient User Identification and Key Distribution Scheme Providing Enhanced Security. *Computer & Security*. Vol. 23. No. 8. (2004) 697-704
7. Lin, C.L., Hwang, T.: A Password Authentication Scheme with Secure Password Updating. *Computers & Security*. Vol. 22. No. 1. (2003) 68-72
8. Yang, C.C., Chang, T.Y., Li, J.W.: Security Enhancement for Protecting Password Transmission. *IEICE Transactions on Communications*. Vol. E86-B. No. 7. (July 2003) 2178-2181
9. Menezes, A.J., Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press. New York. (1997)

# Enhanced ID-Based Authenticated Key Agreement Protocols for a Multiple Independent PKG Environment\*

Sangjin Kim<sup>1</sup>, Hoonjung Lee<sup>2</sup>, and Heekuck Oh<sup>3</sup>

<sup>1</sup> Korea University of Technology and Education,  
School of Information and Media Engineering, Republic of Korea  
sangjin@kut.ac.kr

<sup>2</sup> HANDAN BroadInfoCom, Republic of Korea  
hjlee@handan.co.kr

<sup>3</sup> Hanyang University, Department of Computer Science and Engineering, Republic of Korea  
hkoh@cse.hanyang.ac.kr

**Abstract.** In 2005, Lee et al. proposed an ID-based 2-party key agreement protocol between users whose private keys were issued by independent PKGs that do not share any system parameters. This work was the first kind that assumes completely independent multiple PKG environment. However, Lee et al. protocol has a flaw that allows attackers to impersonate others without knowing their private keys. In this paper, we propose a modification to the protocol of Lee et al. that prevents impersonation attacks. We also show a simple technique that can improve the efficiency of tripartite key agreement protocol of Lee et al. We also provide analysis of the security and efficiency of the proposed protocols.

**Keywords:** ID-based cryptosystem, key agreement protocol, multiple PKG environment.

## 1 Introduction

Key establishment protocols are widely used to share a common secret key between entities. This secret key is normally used as a session key to construct a secure channel between the entities involved. Key establishment protocols can be subdivided into key transport protocols and key agreement protocols. In a key transport protocol, one of the participants creates the shared key and distributes it to others securely. On the other hand, in a key agreement protocol, each entity computes the common secret key using the information contributed by all the entities involved. In this paper, we are concerned with key agreement protocols. The famous Diffie and Hellman [1] key agreement protocol suffered from the man-in-the-middle-attack, which is due to the fact that entities involved are not authenticated. In short, an authenticated key agreement protocol is denoted as *AK protocol*.

In 1984, Shamir introduced the concept of ID-based public key cryptosystem [2]. In these systems, public keys of users' are derived from their well-known unique identity

---

\* This work was supported by the Ministry of Information and Communication, Korea, under the HNRC-ITRC support program supervised by the IITA.



information such as an email address. In ID-based cryptosystem, a trusted authority called the PKG (Private Key Generator) generates user's private keys using the master key of the system. Therefore, the PKG can inherently decrypt any ciphertext or forge signatures of any users'. To overcome this problem, many schemes use multiple PKGs in a threshold manner. In 2001, Boneh and Franklin proposed a practical ID-based encryption scheme based on the Weil pairing [3]. Since then, most researches on ID-based cryptosystem are based on pairings.

ID-based 2-party AK protocol was first proposed by Smart in 2001 [4]. This protocol is based on Boneh and Franklin's work and requires two pairing computations by each user to compute the session key. Smart's protocol, however, does not satisfy the PKG forward secrecy. In 2003, Chen and Kudla [5] introduced a new ID-based 2-party AK protocol that provided the PKG forward secrecy and reduced the pairing computation done by each user to one. In the same paper, they also extended Smart's protocol to a multiple PKG environment where users who acquired their private keys from different PKGs can share a common key. In this protocol, although every PKG uses distinct master key, they share common system parameters. In 2005, Lee et al. proposed ID-based 2-party AK protocol for a multiple independent PKG environment, where PKGs do not share common system parameters [6]. This protocol, however, has a serious flaw that allows attackers to impersonate others freely. Research on tripartite AK protocol was initiated by Joux in 2000 [7]. A first ID-based tripartite AK protocol was introduced by Zhang et al. in 2002 [8]. Shim [9] proposed another ID-based tripartite AK protocol that requires less computation than Zhang et al.'s in 2003. In 2005, Lee et al. proposed an ID-based tripartite AK protocol for a completely independent multiple PKG environment [6]. The protocol, however, uses 2-party protocol that has a flaw.

In ID-based cryptosystems, users acquire their private keys from the PKG. A single PKG may be responsible for issuing private keys to members of a small-scale organization, but it is unrealistic to assume that a single PKG will be responsible for issuing private keys to members of different organizations. Furthermore, it is also unrealistic to assume that different PKGs will share common system parameters and differ only in the master key as assumed by Chen and Kudla [5]. Some argue that standardized system parameters can be shared by distinct PKGs. This assumption, however, is still too limited. Therefore, in this paper, we consider a completely independent multiple PKG environment, where all the PKGs use different system parameters.

To date, most of the ID-based AK protocols are based on a single PKG environment [4, 8, 9]. As we will explain in section 4, it is not fairly straightforward to extend these protocols to a setting where multiple independent PKGs exist. In 2005, Lee et al. proposed ID-based 2-party and tripartite AK protocols for this setting [6]. However, there is a critical flaw that allows attackers to impersonate others freely. In this paper, we propose a modification to the protocol of Lee et al. that preserves the efficiency but removes the flaw of the original protocol. We also show a simple way of improving the efficiency of tripartite AK protocol of Lee et al. The proposed 2-party and tripartite AK protocols requires only two and four pairing computations for each users, respectively. Therefore, only one more additional pairing computation is required compared to the most efficient protocols for a single PKG environment.

## 2 Security Attributes of Key Agreement Protocol

The followings are the security requirements of key agreement protocols, some of which are specific to only ID-based AK protocols.

- **Known-key security:** Each run of the key agreement protocol should generate a unique and independent session key. An adversary must have negligible advantage on compromising future session keys, even though it has compromised past session keys.
- **Forward secrecy:** An adversary must have negligible advantage on compromising past session keys, even though it has compromised long-term private keys of one or more participants. The notion of forward secrecy can be further extended to the following two types of secrecy.
  - **Perfect forward secrecy:** The forward secrecy must be preserved even if long-term private keys of all the participants involved are compromised.
  - **PKG forward secrecy:** The forward secrecy must be preserved even if the master key of the PKG is compromised.
- **Key-compromise resilience:** An adversary must have negligible advantage on impersonating others to  $A$ , even if it has compromised  $A$ 's private key.
- **Unknown key share resilience:** An adversary must have negligible advantage on coercing others into sharing a key with other entities when it is actually sharing with a different entity.
- **Key control:** An adversary must have negligible advantage on forcing the session key to be a pre-selected value.

## 3 Mathematical Background

From now on, we will use the following notations: 1)  $q$  is a large prime number, 2)  $\mathbb{G}_1$  and  $\mathbb{G}_2$  are two groups with the same order  $q$ , where  $\mathbb{G}_1$  is an additive group on an elliptic curve, and  $\mathbb{G}_2$  is a multiplicative group of a finite field, 3)  $P$ ,  $Q$ , and  $R$  are random elements of  $\mathbb{G}_1$ , and 4)  $a$ ,  $b$ , and  $c$  are random elements of  $\mathbb{Z}_q^*$ .

**Definition 1 (Admissible Bilinear Map).** A map  $\hat{e} : G_1 \times G_1 \rightarrow G_2$  is an admissible bilinear map if and only if it satisfies the following properties.

- **Bilinear:** Given  $P$ ,  $Q$ , and  $R$ , the followings hold.
  - $\hat{e}(P + Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R)$
  - $\hat{e}(P, Q + R) = \hat{e}(P, Q) \cdot \hat{e}(P, R)$
- **Non-degenerate:** The map does not send all pairs in  $\mathbb{G}_1 \times \mathbb{G}_1$  to the identity in  $\mathbb{G}_2$ .
- **Computable:** There exists an efficient algorithm to compute  $\hat{e}(P, Q)$  for all  $P, Q \in \mathbb{G}_1$

**Definition 2 (Discrete Logarithm Problem (DLP) in  $\mathbb{G}_1$ ).** DLP is as follows: Given  $\langle P, aP \rangle$ , compute  $a \in \mathbb{Z}_q$ .

**Definition 3 (Computational Diffie-Hellman Problem (CDHP) in  $\mathbb{G}_1$ ).** CDHP is as follows: Given  $\langle P, aP, bP \rangle$ , compute  $abP \in \mathbb{G}_1$ .

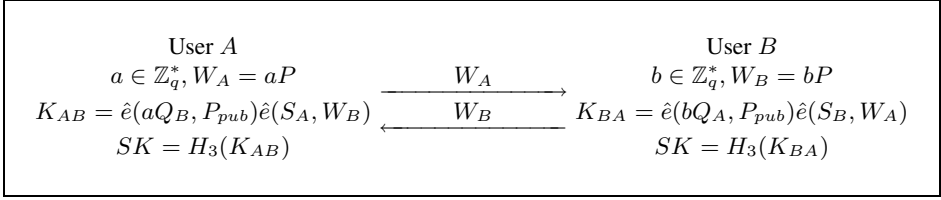


Fig. 1. Smart’s 2-Party ID-based AK Protocol

**Definition 4 (Bilinear Diffie-Hellman Problem (BDHP) in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ ).** *BDHP is as follows: Given  $\langle P, aP, bP, cP \rangle$ , compute  $\hat{e}(P, P)^{abc} \in \mathbb{G}_2$ .*

Currently, solving DLP, CDHP, and BDHP is computationally infeasible. For more detail, refer to [3].

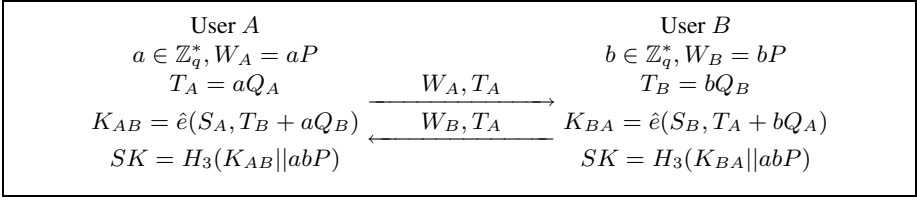
#### 4 Difficulty of Designing ID-Based AK Protocols for a Multiple PKG Environment

In this section, we will use the following notations additionally to the notations defined in section 3. 1)  $g$  is a generator of  $\mathbb{G}_2$ . 2) The PKG’s master key is  $s \in \mathbb{Z}_q^*$  and the corresponding public key is  $P_{pub} = sP$ . 3)  $ID_A$  denotes the identity of the user  $A$ . 4)  $Q_A = H_1(ID_A)$  is the public key of the user  $A$  and  $S_A = sQ_A$  is the corresponding private key, where  $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$  is a collision-resistant hash function. 5)  $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$  and  $H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^k$  are also collision-resistant hash functions.

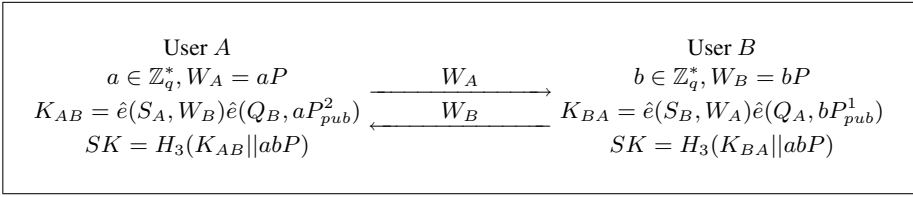
Diffie-Hellman key agreement protocol can be easily converted to an elliptic curve version by exchanging  $aP, bP$  and using  $abP$  as the session key. However, this naive version also suffers from the man-in-the-middle attack. To counter this problem, user  $A$  could send  $V_A = H_2(W_A)S_A + aP_{pub}$ , which is a digital signature on  $W_A = aP$  to user  $B$ , with  $W_A$ .  $B$  can verify  $V_A$  by testing whether  $\hat{e}(V_A, P)$  equals  $\hat{e}(H_2(W_A)Q_A + W_A, P_{pub})$ . This kind of protocol requires two pairing computations per each user and the bandwidth of each message is two elliptic curve points.

There is an another way to counter the man-in-the-middle attack which do not require digitally signing the values exchanged. In this method, the session key is generated in a way that only the legitimate users can create it. Fig 1 shows the Smart’s protocol that uses such method [4]. This protocol requires the same amount of pairing computation as the protocol using signatures. However, it is more efficient with respect to the message bandwidth. The drawback of this protocol is that the PKG can always compute the session key using  $K_{AB} = \hat{e}(S_A, W_B)\hat{e}(S_B, W_A)$ .

Chen and Kudla [5] proposed an ID-based AK protocol depicted in Fig 2. This protocol requires only a single pairing computation per each participant and provides the PKG forward secrecy. However, two elliptic curve points are exchanged compared to one in Smart’s. The PKG forward secrecy is satisfied by computing the session key  $SK$  using the additional input  $abP$ . Although, the PKG can compute the other input using  $K_{AB} = \hat{e}(Q_A, W_B)^s\hat{e}(Q_B, W_A)^s$ , it cannot compute  $abP$  due to the infeasibility of



**Fig. 2.** Chen and Kudla's 2-Party AK Protocol for a single PKG environment

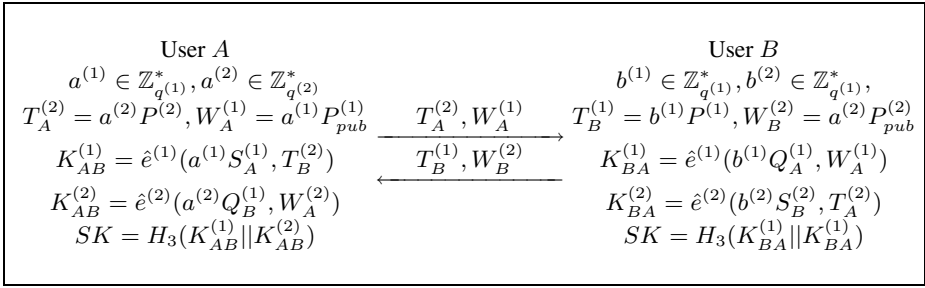


**Fig. 3.** Chen and Kudla's 2-Party AK Protocol for a multiple PKG environment

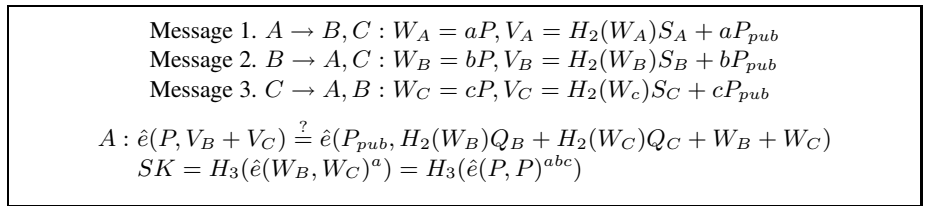
CDHP. They also proposed a 2-party protocol for a multiple PKG environment shown in Fig 3. They, however, assumed that PKGs share system parameters and differ only in their master keys. In this protocol, the master key of PKG<sub>1</sub> and PKG<sub>2</sub> are  $s_1 \in \mathbb{Z}_q^*$  and  $s_2 \in \mathbb{Z}_q^*$ , and the corresponding public keys are  $P_{pub}^1 = s_1P$  and  $P_{pub}^2 = s_2P$ , respectively. The user  $A$  and  $B$  acquired his/her private key  $S_A = s_1Q_A$  and  $S_B = s_2Q_B$  from PKG<sub>1</sub> and PKG<sub>2</sub>, respectively. If the PKG<sub>1</sub> and PKG<sub>2</sub> do not share system parameters, we cannot directly apply this protocol due to the fact that the order of the group used by each environment is different.

In 2005, Lee et al. proposed an ID-based 2-party AK protocol depicted in Fig 4 for a multiple independent PKG environment [6]. The superscript (1) and (2) denotes that the values are suitable for the environment of the PKG<sub>1</sub> and PKG<sub>2</sub>, respectively. If the both parties follow the protocol, no attack on this protocol is possible as was argued in their paper. However, an attacker who does not know the private key of  $A$  can impersonate  $A$  by sending  $W_A^{(1)} = a^{(1)}P^{(1)}$  instead of  $W_A^{(1)} = a^{(1)}P_{pub}^{(1)}$ . In this case, the two input values of the session key becomes  $K_{BA}^{(1)} = \hat{e}^{(1)}(Q_A^{(1)}, P^{(1)})^{a^{(1)}b^{(1)}}$  and  $K_{BA}^{(2)} = \hat{e}^{(2)}(Q_B^{(1)}, P^{(2)})^{a^{(2)}b^{(2)}s^{(2)}}$ , which can be computed by the attacker. This attack is possible due to the fact that  $B$  cannot verify the structure of the values exchanged.

Using a multiplicative group of a finite field, it is difficult for three parties to agree on a key of the form  $g^{abc}$ . However, using the bilinear property of a bilinear map, it is straightforward for three parties to agree on a key of the form  $g^{abc}$ . For example, as proposed by Joux [7], we can construct a tripartite key agreement protocol by exchanging  $aP$ ,  $bP$ , and  $cP$ , and use  $\hat{e}(bP, cP)^a = \hat{e}(P, P)^{abc}$  as the session key. However, since the values exchanged are not authenticated, the protocol will also be susceptible to the man-in-the-middle attack. To counter this attack, we could digitally sign the values exchanged or apply the Smart's approach. However, in the latter case, it is difficult to



**Fig. 4.** Lee et al. 2-Party Key Agreement Protocol for a multiple PKG environment



**Fig. 5.** Shim’s Tripartite Key Agreement Protocol

devise a pairing equation that can be computed only by the three legitimate participants. In the former case, it requires basically 4 pairing computation to verify signatures on two values, and one more pairing computation to compute the session key. As a result, total 5 pairing operations is needed.

In a single PKG environment, we can reduce the pairing computations required to verify the signature of two values in a single pairing equation as was done by Shim [9]. The protocol proposed by Shim is depicted in Fig 5. We only show how  $A$  computes his/her session key. This reduction technique, however, cannot be applied to multiple PKG environment, where PKGs do not share the system parameters. As a result, in such environment, the minimum pairing computation required will be five per each user. We could also think of using the 2-party AK protocols in tripartite AK protocols. For example, we could first run 2-party AK protocols between  $A$  and  $B$ ,  $A$  and  $C$ , and  $B$  and  $C$ . Then combine the resulting three keys into a single key. However,  $A$  cannot obtain the shared key between  $B$  and  $C$ , if it is not explicitly sent to him/her. Therefore, two runs are required to use this approach. Lee et al. used this approach [6].

## 5 The Enhanced Protocols

### 5.1 System Setup

The system setup phase is similar to that of Lee et al. [6]. The  $n$  different PKGs, which do not share common system parameters configure their parameters as follows.

- $\text{PKG}_i$  chooses its basic system parameter:  $\langle \mathbb{G}_1^{(i)}, \mathbb{G}_2^{(i)}, \hat{e}^{(i)} \rangle$ , where  $\mathbb{G}_1^{(i)}$  is an additive group of order  $q^{(i)}$ ,  $\mathbb{G}_2^{(i)}$  is a multiplicative group of the same order  $q^{(i)}$ , and  $\hat{e}^{(i)}$  is admissible bilinear map between  $\mathbb{G}_1^{(i)}$  and  $\mathbb{G}_2^{(i)}$ .
- $\text{PKG}_i$  chooses  $P^{(i)}$ , a random generator of  $\mathbb{G}_1^{(i)}$ . It also chooses a collision-resistant hash functions  $H_1^{(i)} : \{0, 1\}^* \rightarrow \mathbb{G}_1^{(i)}$ .
- Finally,  $\text{PKG}_i$  randomly chooses its master key  $s^{(i)} \in \mathbb{Z}_{q^{(i)}}^*$ . It also computes the corresponding public key  $P_{\text{pub}}^{(i)} = s^{(i)} P^{(i)}$ .

After completing the system setup phase, each PKG publishes its public system parameters:  $\langle q^{(i)}, \mathbb{G}_1^{(i)}, \mathbb{G}_2^{(i)}, P^{(i)}, P_{\text{pub}}^{(i)}, H_1^{(i)}, \hat{e}^{(i)} \rangle$ . We assume that all users agree on the hash function  $H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^k$  used to compute the resulting session key, where  $k$  is the length of the session key.

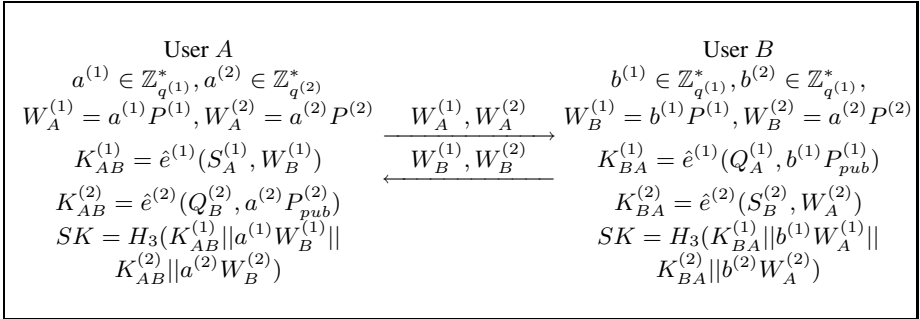


Fig. 6. The 2PAK-MPE Protocol

### 5.2 The 2PAK-MPE Protocol

In this section, we will introduce a new ID-based 2-party key agreement protocol which removes the flaw in Lee et al. [6] protocol. We will refer to this protocol as 2PAK-MPE(2-Party Authenticated Key agreement protocol for Multiple independent PKG Environment). This protocol is performed between two entities  $A$  and  $B$  who have acquired their private key from  $\text{PKG}_1$  and  $\text{PKG}_2$ , respectively. For example, the public key of  $A$  is  $Q_A^{(1)} = H_1^{(1)}(\text{ID}_A)$  and the corresponding private key is  $S_A^{(1)} = s^{(1)} Q_A^{(1)}$ , where  $\text{ID}_A$  is the identity of  $A$ . We assume that there is an efficient way to acquire authenticated system parameters of  $\text{PKG}_i$  and users knows the other party's ID and system parameters before running the protocol. Our 2PAK-MPE protocol is depicted in Fig 6. The  $W_A^{(1)}, W_A^{(2)}$  of  $A$ ,  $W_B^{(1)}, W_B^{(2)}$  of  $B$  can be pre-computed before the protocol run. The inclusion of  $a^{(1)} W_B^{(1)}, a^{(2)} W_B^{(2)}, b^{(1)} W_A^{(1)}$ , and  $b^{(2)} W_A^{(2)}$  in  $SK$  is to provide the PKG forward secrecy. This idea is from Chen and Kudla's [5]. We can show that both participant have agreed on the same session key  $SK$  by the followings:

Message 1.  $A \rightarrow B, C : W_A^{(1)} = a^{(1)} P^{(1)}, W_A^{(2)} = a^{(2)} P^{(2)}, W_A^{(3)} = a^{(3)} P^{(3)}$   
 Message 2.  $B \rightarrow A, C : W_B^{(1)} = b^{(1)} P^{(1)}, W_B^{(2)} = b^{(2)} P^{(2)}, W_B^{(3)} = b^{(3)} P^{(3)}$   
 Message 3.  $C \rightarrow A, B : W_C^{(1)} = c^{(1)} P^{(1)}, W_C^{(2)} = c^{(2)} P^{(2)}, W_C^{(3)} = c^{(3)} P^{(3)}$

**Fig. 7.** The First Round of 3PAK-MPE

$$\begin{aligned}
 K_{AB}^{(1)} &= \hat{e}^{(1)}(S_A^{(1)}, W_B^{(1)}) \\
 &= \hat{e}^{(1)}(s^{(1)} Q_A^{(1)}, b^{(1)} P^{(1)}) \\
 &= \hat{e}^{(1)}(Q_A^{(1)}, P^{(1)})^{s^{(1)} b^{(1)}} \\
 &= \hat{e}^{(1)}(Q_A^{(1)}, b^{(1)} s^{(1)} P^{(1)}) \\
 &= \hat{e}^{(1)}(Q_A^{(1)}, b^{(1)} P_{pub}^{(1)}) \\
 &= K_{BA}^{(1)},
 \end{aligned}
 \qquad
 \begin{aligned}
 K_{AB}^{(2)} &= \hat{e}^{(2)}(Q_B^{(2)}, a^{(2)} P_{pub}^{(2)}) \\
 &= \hat{e}^{(2)}(Q_B^{(2)}, a^{(2)} s^{(2)} P_{pub}^{(2)}) \\
 &= \hat{e}^{(2)}(Q_B^{(2)}, P^{(2)})^{s^{(2)} a^{(2)}} \\
 &= \hat{e}^{(2)}(s^{(2)} Q_B^{(2)}, a^{(2)} P^{(2)}) \\
 &= \hat{e}^{(2)}(S_B^{(2)}, W_A^{(2)}) \\
 &= K_{BA}^{(2)}.
 \end{aligned}$$

It is clear that this protocol is a role symmetric protocol. A key agreement protocol is referred to as role symmetric if the entities involved executes the same operations.

### 5.3 The 3PAK-MPE Protocol

In this section, we introduce our new ID-based tripartite key agreement protocol called the 3PAK-MPE. We assume that there are three participants  $A$ ,  $B$ , and  $C$ , who have acquired their private key from  $\text{PKG}_1$ ,  $\text{PKG}_2$ , and  $\text{PKG}_3$ , respectively. The notations used here are the same as the ones used in describing the 2PAK-MPE. The protocol is divided into two discrete rounds. In the first round, each entity constructs separate secure and authenticated channels between each other. To achieve this goal, every entity performs 2PAK-MPE with each other individually. We use this method to exploit our 2PAK-MPE protocol and to reduce the required computation while sacrificing the required bandwidth. In the second round, each entity exchanges contributions that are used to compute the session key. These contributions are exchanged in a ciphertext constructed using the key obtain from running 2PAK-MPE.

**The First Round.** In this protocol, each user chooses three ephemeral key from each environment. For example,  $A$  chooses  $a^{(1)} \in \mathbb{Z}_{q^{(1)}}^*$ ,  $a^{(2)} \in \mathbb{Z}_{q^{(2)}}^*$ , and  $a^{(3)} \in \mathbb{Z}_{q^{(3)}}^*$ . The messages exchanged in the first round is depicted in Fig 7. After exchanging messages depicted in Fig 7, each entity computes the partial session keys. In detail,  $A$  computes partial keys  $K_{AB}$  and  $K_{AC}$ , which are used to construct a secure channel between  $A$  and  $B$  and  $A$  and  $C$ , respectively, as follows:

$$\begin{aligned}
 K_{AB} &= H_3(\hat{e}^{(1)}(S_A^{(1)}, W_B^{(1)}) || a^{(1)} W_B^{(1)} || \hat{e}^{(2)}(Q_B^{(2)}, a^{(2)} P_{pub}^{(2)}) || a^{(2)} W_B^{(2)}), \\
 K_{AC} &= H_3(\hat{e}^{(1)}(S_A^{(1)}, W_C^{(1)}) || a^{(1)} W_C^{(1)} || \hat{e}^{(3)}(Q_C^{(3)}, a^{(3)} P_{pub}^{(3)}) || a^{(3)} W_C^{(3)}).
 \end{aligned}$$

Similarly,  $B$  and  $C$  also computes its partial session keys as follows:

$$\begin{aligned}
 K_{BA} &= H_3(\hat{e}^{(1)}(Q_A^{(1)}, b^{(1)} P_{pub}^{(1)}) || b^{(1)} W_A^{(1)} || \hat{e}^{(2)}(S_B^{(2)}, W_A^{(2)}) || b^{(2)} W_A^{(2)}), \\
 K_{BC} &= H_3(\hat{e}^{(2)}(S_B^{(2)}, W_C^{(2)}) || b^{(2)} W_C^{(2)} || \hat{e}^{(3)}(Q_C^{(3)}, b^{(3)} P_{pub}^{(3)}) || b^{(3)} W_C^{(3)}),
 \end{aligned}$$

Message 1.  $A \rightarrow B, C : \{R_A\}_{K_{AB}}, \{R_A\}_{K_{AC}}$   
 Message 2.  $B \rightarrow A, C : \{R_B\}_{K_{BA}}, \{R_B\}_{K_{BC}}$   
 Message 3.  $C \rightarrow A, B : \{R_C\}_{K_{CA}}, \{R_C\}_{K_{CB}}$

**Fig. 8.** The Second Round of 3PAK-MPE

$$K_{CA} = H_3(\hat{e}^{(1)}(Q_A^{(1)}, c^{(1)} P_{pub}^{(1)}) || c^{(1)} W_A^{(1)} || \hat{e}^{(3)}(S_C^{(3)}, W_A^{(3)}) || c^{(3)} W_A^{(3)}),$$

$$K_{CB} = H_3(\hat{e}^{(2)}(Q_B^{(2)}, c^{(2)} P_{pub}^{(2)}) || c^{(2)} W_B^{(2)} || \hat{e}^{(3)}(S_C^{(3)}, W_B^{(3)}) || c^{(3)} W_B^{(3)}).$$

At the end of this round, each entity obtains two session keys that can be used between other entities involved.

**The Second Round.** In the second round, each entity exchanges the contributions that will be used to compute the final session key. The message exchanged in this round is given in Fig 8. Here,  $R_A$ ,  $R_B$ , and  $R_C$  denotes nonces chosen by each user, and  $\{M\}_K$  denotes encryption of  $M$  using the symmetric key  $K$ . Since the values are exchanged with a shared key, each user can be confident that these values are from the corresponding users. We could also include some redundancy in the encrypted message, for example  $\{R_A || Q_A || Q_B\}_{K_{AB}}$ , to allow each user to verify that the correct keys were used in each encryption. After exchanging the messages,  $A$ ,  $B$ , and  $C$  computes the session key as  $SK = H_3(R_A || R_B || R_C)$ . Generally, a session key resulting from a tripartite AK protocol using pairing is  $\hat{e}(P, P)^{abc}$ . We deliberately did not use this form to reduce the required number of pairing computation. If all three entities are from different environment, it may be difficult to agree on the common  $P$ .

## 6 Analysis

In this section, we analyze the security and the efficiency of the proposed protocols. We first heuristically argue that our protocols satisfy the security requirements of the AK protocols. We then discuss the efficiency of our protocols by comparing the number of pairing computations required with other ID-based AK protocols.

### 6.1 Security Analysis

We only discuss the security of 2PAK-MPE protocol. Since the 3PAK-MPE uses the 2PAK-MPE, the security of 3PAK-MPE depends on 2PAK-MPE. If 2PAK-MPE protocol is secure, it is computationally infeasible for an attacker to obtain the contributions, which are exchanged encrypted using the keys obtained from 2PAK-MPE, used to compute the final session key.

**Lemma 1.** *Only the legitimate participants of 2PAK-MPE can compute the resulting session key of 2PAK-MPE.*



*Proof.* The publicly available values are as follows:

$$Q_A^{(1)}, Q_B^{(2)}, s^{(1)}P^{(1)}, s^{(2)}P^{(2)}, a^{(1)}P^{(1)}, a^{(2)}P^{(2)}, b^{(1)}P^{(1)}, \text{ and } b^{(2)}P^{(2)}.$$

We can subdivide the possible attackers into passive attackers and active attackers. We will first consider the following four types of passive attackers.

- **Type 1.** This type of attacker only knows the publicly available information. To compute  $K_{AB}^{(1)}$ , the attacker needs to obtain  $b^{(1)}$  from  $b^{(1)}P^{(1)}$ ,  $s^{(1)}$  from  $s^{(1)}P^{(1)}$ ,  $S_A^{(1)}$ , or compute  $b^{(1)}P_{pub}^{(1)}$  from  $b^{(1)}P^{(1)}$  and  $s^{(1)}P^{(1)}$ . If the attacker can obtain  $b^{(1)}$  or  $s^{(1)}$ , he/she can compute  $K_{AB}^{(1)} = \hat{e}^{(1)}(Q_A^{(1)}, P_{pub}^{(1)})^{b^{(1)}} = \hat{e}^{(1)}(Q_A^{(1)}, b^{(1)}P^{(1)})^{s^{(1)}}$ . However, obtaining these values are computationally infeasible due to the DLP in  $\mathbb{G}_1$ . Furthermore, computing  $b^{(1)}P_{pub}^{(1)}$  is computationally infeasible due to the CDHP in  $\mathbb{G}_1$ .
- **Type 2.** This type of attacker knows the master key of both PKGs and the publicly available information. This means the attacker knows both  $s^{(1)}$  and  $s^{(2)}$ . As a result, this type of attacker can compute  $K_{AB}^{(1)} = \hat{e}^{(1)}(Q_A^{(1)}, b^{(1)}P^{(1)})^{s^{(1)}}$  and  $K_{AB}^{(2)} = \hat{e}^{(1)}(Q_B^{(2)}, a^{(2)}P^{(2)})^{s^{(2)}}$ . However, to compute the final session key, the attacker must be able to compute  $a^{(1)}W_B^{(1)}$  or  $b^{(1)}W_A^{(1)}$  and  $a^{(2)}W_B^{(2)}$  or  $b^{(2)}W_A^{(2)}$ . Computing these values are computationally infeasible due to the CDHP in  $\mathbb{G}_1$ .
- **Type 3.** This type of attacker knows the ephemeral key of one of the participants and the publicly available information. For example, if the attacker, knows  $a^{(1)}$  and  $a^{(2)}$ , he/she can compute  $K_{AB}^{(2)} = \hat{e}^{(2)}(Q_B^{(2)}, P_{pub}^{(2)})^{a^{(2)}}$ . However, this attacker needs to obtain  $b^{(1)}$  from  $b^{(1)}P^{(1)}$ ,  $s^{(1)}$  from  $s^{(1)}P^{(1)}$ ,  $S_A^{(1)}$ , or compute  $b^{(1)}P_{pub}^{(1)}$  from  $b^{(1)}P^{(1)}$  and  $s^{(1)}P^{(1)}$  to compute  $K_{AB}^{(1)}$ . This is identical situation to an attacker of type 1.
- **Type 4.** This type of attacker knows the private key of one of the participants as well as the ephemeral key of the opposite participant. Let's assume the attacker knows  $S_A^{(1)}$ ,  $b^{(1)}$ , and  $b^{(2)}$  in addition to the publicly available information. This attacker can obviously compute  $K_{AB}^{(1)} = \hat{e}^{(1)}(S_A^{(1)}, W_B^{(1)})$ . However, with respect to computing  $K_{AB}^{(2)}$ , the additional values  $S_A^{(1)}$ ,  $b^{(1)}$ , and  $b^{(2)}$  does not help the attacker in any way. The attacker needs  $a^{(2)}$ ,  $s^{(2)}$ ,  $a^{(2)}P_{pub}^{(2)}$  or  $S_B^{(2)}$  to compute this value. Obtaining the first two values and the third value are computationally infeasible due to DLP in  $\mathbb{G}_1$  and CDHP in  $\mathbb{G}_1$ , respectively.

Now, we will consider active attackers. This type of attacker can alter the message and may not follow the protocol specification. One possible attack is sending  $P^{(i)}$ ,  $a^{(i)}P_{pub}^{(i)}$ , or  $a^{(i)}Q_A^{(i)}$  instead of the original form  $a^{(i)}P^{(i)}$ . Since the recipient cannot verify the form, this attack will not be detected. However, since  $K_{BA}^{(1)} = \hat{e}^{(1)}(Q_A^{(1)}, b^{(1)}P_{pub}^{(1)})$  is computed without using any of the values received, this attack does not affect  $K_{BA}^{(1)}$ . Moreover, as argued in passive attacker of type 1, it is computationally infeasible for an active attacker to compute  $K_{BA}^{(1)}$  using only the publicly available information. Therefore, an active attacker cannot share a key with a legitimate user by modifying the values

exchanged. From these arguments, this protocol is secure with respect to both passive and active attackers.  $\square$

1. **Known-key security:** In our protocol, ephemeral keys such as  $a^{(i)}$  and  $b^{(i)}$  are used to construct the session key. As a result, each run of the protocol creates unique session key that is independent of past or future session keys. Therefore, compromise of past session keys do not affect the security of future session keys.
2. **PKG forward secrecy:** To satisfy PKG forward secrecy, the compromise of master keys of  $PKG_1$  and  $PKG_2$  must not affect the security of past session keys. This secrecy corresponds to a passive attacker of type 2 discussed in Lemma 1. As a result, it is computationally infeasible to compute the resulting session key, even if an attacker obtains the master keys of both PKGs.
3. **Key-compromise resilience:** This resilience corresponds to a passive attacker of type 4 and active attackers discussed in Lemma 1. As a result, it is computationally infeasible for an attacker to impersonate others to  $A$ , even if the private key of  $A$  is known to the attacker.
4. **Unknown key-share resilience:** This resilience corresponds to a passive attacker of type 4 and active attackers discussed in Lemma 1. As a result, it is computationally infeasible for an attacker to deceive a party into falsely believing the identity of the opposite party in concern.
5. **Key control:** Since each party contributes a fresh ephemeral key as one of the input used to compute the session key, one of the party cannot force the session key to be some pre-selected value.
6. **Man-in-the-middle-attack:** If an attacker intercepts the two messages and sends  $W_C^{(1)} = c^{(1)}P^{(1)}$ ,  $W_C^{(2)} = c^{(2)}P^2$  to  $A$ , the computed partial key will be as follows:

$$\begin{aligned}
 K_{AC}^{(1)} &= \hat{e}^{(1)}(S_A^{(1)}, W_C^{(1)}) = \hat{e}^{(1)}(Q_A^{(1)}, P^{(1)})^{c^{(1)}s^{(1)}}, \\
 K_{AC}^{(2)} &= \hat{e}^{(2)}(Q_B^{(2)}, a^{(2)}P_{pub}^{(2)}) = \hat{e}^{(2)}(Q_B^{(2)}, P^{(2)})^{a^{(2)}s^{(2)}}.
 \end{aligned}$$

This type of attacker has the same amount of information as a passive attacker of type 3 discussed in Lemma 1. As a result, although the attacker can compute  $K_{AC}^{(1)} = \hat{e}^{(1)}(Q_A^{(1)}, P_{pub}^{(1)})^{c^{(1)}}$ , it is computationally infeasible for the attacker to compute  $K_{AC}^{(2)}$ .

## 6.2 Efficiency Analysis

In this subsection, since pairing computation out weigh other computations, we first compare our protocol with others using the number of pairing computations required. In Table 1, we compare our 2PAK-MPE with Chen and Kudla’s protocol given in Fig 2 and Fig 3. We can see that the efficiency of our protocol is equal to Chen and Kudla’s even though in our setting each PKGs uses different system parameters. In Table 2, we compare our 3PAK-MPE protocol with Shim’s protocol given in Fig 5 and a hypothetical protocol for a multiple PKG environment. In the hypothetical protocol, we assume that the users have agree on which  $P^{(i)}$  to use in advance. Our protocol requires only one more computation than Shim’s even though our protocol is for a multiple PKG environment. Furthermore, it is more efficient than the hypothetical protocol. With respect

**Table 1.** Comparison of pairing computation in two-party AK protocol

protocol	single PKG environment		multiple PKG environment	
	pairing(each)	pairing(all)	pairing(each)	pairing(all)
Chen and Kudla’s protocol	1	2	2*	4
2PAK-MPE protocol			2**	4

\*: each PKG shares the common system parameters but has distinct master key.

\*\* : each PKG uses different system parameters.

**Table 2.** Comparison of pairing computation in tripartite key agreement protocol

protocol	single PKG environment		multiple PKG environment	
	pairing(each)	pairing(all)	pairing(each)	pairing(all)
Shim’s protocol	3	9		
a hypothetical protocol*			5	15
3PAK-MPE protocol			4	12

\*: a direct extension of Shim’s to a multiple independent PKG environment with the final session key of the form  $\hat{e}^{(i)}(P^{(i)}, P^{(i)})^{a^{(i)} b^{(i)} c^{(i)}}$ .

to message bandwidth, both 2PAK-MPE and Chen and Kudla’s exchanges two elliptic curve points per message. 3PAK-MPE requires three elliptic curve points per message whereas four points are exchanged in the hypothetical protocol. However, 3PAK-MPE consists of two separate rounds, whereas only one round is needed in the hypothetical protocol.

## 7 Conclusion

In this paper, we provide a modification to Lee et al.’s ID-based AK protocols for a completely independent multiple PKG environment. The original 2-party AK protocol contains a flaw that allow attackers to impersonate others freely. We believe that we have successfully removed this flaw, while preserving the efficiency of the original. We have also showed a very simple way to reduce the amount of pairing computations required in tripartite AK protocols. As a result, the efficiency of our protocols are similar to previous ID-based AK protocols for a single PKG environment. Lastly, we have provided a more detail analysis of security of the proposed protocols.

## References

1. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Trans. on Information Theory, Vol. 22, No. 6. IEEE Press (1976) 664–654
2. Shamir, A.: Identity-based Cryptosystems and Signature Schemes. Advances in Cryptology, Crypto 1984. Lecture Notes in Computer Science, Vol. 196. Springer (1985) 47–53

3. Boneh, D., Franklin, M.: Identity-based Encryption from the Weil pairing. *Advances in Cryptology, Crypto 2001*. Lecture Notes in Computer Science, Vol. 2139. Springer (2001) 213–229
4. Smart, N.: An Identity-based Authenticated Key Agreement Protocol Based on Weil Pairing. *Electronic Letters*, Vol. 38, No. 13. IEE Press(2002) 630–632
5. Chen, L., Kudla, C.: Identity-based Authenticated Key Agreement Protocols from Pairings. *Proc. of the 16th IEEE Computer Security Foundations Workshop*. IEEE Press (2003) 219–233
6. Lee, H., Kim, D., Kim, S., Oh, H.: Identity-based Key Agreement Protocols in a Multiple PKG Environment. *Proc. of the Int. Conf. on Computational Science and Its Applications, ICCSA 2005*. Lecture Notes in Computer Science, Vol. 3483. Springer (2005) 877–886
7. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. *Proc. of Algorithmic Number Theory Symp., ANTS-IV*. Lecture Notes in Computer Science, Vol. 1838. Spinger (2000) 385–394
8. Zhang, F., Liu, S., Kim, K.: ID-Based One Round Authenticated Tripartite Key Agreement Protocols with Pairings. *IACR Cryptology ePrint Archive, Report 2002/122*. (2002)
9. Shim, K.: Cryptanalysis of ID-based Tripartite Authenticated Key Agreement Protocols. *IACR Cryptology ePrint Archive, Report 2003/115*. (2003)

# Enforce Mandatory Access Control Policy on XML Documents

Lan Li, Xinghao Jiang, and Jianhua Li

School of Information Security Engineering, Shanghai Jiao Tong University,  
Shanghai, 200030, China

Key Lab of Integrate Administration Technologies for Information Security,  
Shanghai, 200030, China

{lanli, xhjjiang, lijh888}@sjtu.edu.cn

**Abstract.** Information stored in XML documents should be protected from unauthorized access. In military or other highly secure environments, mandatory access control (MAC) policy should be enforced on the sensitive information. If we use XML documents to store or exchange information in these environments, we should also enforce MAC policy on these XML documents. In this paper, we discussed a method to enforce fine-grained MAC policy on XML documents. The model of XML document is extended to contain the security information – label. Three kinds of labels are defined to determine the labels of the nodes in XML documents. Security view of XML document under MAC policy is proposed in this paper. The operations on XML documents will be redirected to the security views which contain the proper nodes under MAC policy. Validity of the security views is also described. Four kinds of operations on XML documents are discussed in details to explain how to enforce mandatory access control. The problem of polyinstantiation caused by these operations is also discussed. At last the architecture of enforcing MAC policy on XML documents is presented.

## 1 Introduction

XML (Extensible Markup Language [16]) documents have been used to store and exchange information in many environments. The management of XML data is becoming an important problem when there are many XML documents in the system. Especially the security of XML data plays an important role in XML data management. Access control is one of the methods to guarantee the data's security.

In military or highly secure environments, such as the governments and other huge organizations, Mandatory Access Control (MAC) has been used to enforce strict security policy on data. MAC policy controls the authorization of data according to the users and objects' security attributes which common users can not modify. In contrast, Discretionary Access Control (DAC) allows users to manage the authorization of their own objects. DAC cannot resist the attacks such as Trojan horse because of its inherent flaws. MAC policy controls all the information flows of the system to avoid the attacks of Trojan horse. Bell-La Padula (BLP) model [3] is a frequently used MAC model in many military and highly secure systems. Labels are used to represent the security attributes of the users and objects. If BLP model is applied in XML environments, the

XML document model must be modified to contain the labels. To enforce a fine-grained access control policy on XML documents, the elements and attributes in the document are the objects and they all should have labels. Thus different nodes of an XML document may have different labels. A user can only execute operations on certain nodes according to MAC policy. We will define three kinds of labels for the objects to make the management easier. The system can determine the label of an object by the label assigned to it, the label assigned to its definition or the label of its container.

Security view is one reasonable method to implement the access control in database systems and XML-based systems. We will define our security view in this paper. What a user accesses is not the original document but a security view that only contains parts of the document constrained by MAC policy. Although many XML-based systems do not have schema files to validate their documents, more and more systems will process only valid XML documents in the future. We will discuss the validity of XML security views under MAC policy. Operation executor will redirect users' operations to the proper security views of the documents. We discuss four kinds of operations: query, insert, delete and modify. The updated contents of a security view will be reflected to the original document after the successful execution of an update operation. Like multilevel database, polyinstantiation also emerges after some update operations.

The architecture of enforcing MAC policy on XML documents will be presented in this paper. The XML technologies can be used in most parts of our architecture. The existed XML-based systems are easy to be extended to enforce MAC policy because the nodes' labels are stored outside the original XML documents. Security view generator is the critical module in the architecture to enforce MAC policy.

The paper is organized as follows: Section 2 introduces the strict BLP model. Section 3 presents modified XML document model containing labels and defines three kinds of labels in XML documents. The security view under MAC policy is proposed in section 4. The validity of security view is also discussed. Section 5 describes the access control on the four kinds of operations under MAC policy. The problem of polyinstantiation is discussed in this section. Section 6 is a description of the architecture of enforcing MAC policy on XML documents. Section 7 is about related work. The last section gives the conclusion of this paper.

## 2 Strict Bell-La Padula Model

Bell-La Padula (BLP) model is the most widely used MAC policy. Subjects ( $S$ ) are the active entities that can access the data. Objects ( $O$ ) are the passive entities that store information and be accessed by subjects. In BLP model, every entity has the security attribute called label. Label is denoted by  $\lambda$ . The label consists of two elements: class ( $C$ ) and category ( $G$ ). Class values from a sequence, and the category is a subset of a set. A label can be represented by a tuple:  $(C, G)$ . A label  $\lambda_1(C_1, G_1)$  dominates another label  $\lambda_2(C_2, G_2)$  iff  $C_1 \geq C_2$  and  $G_1 \supseteq G_2$ , and we denoted it by  $\lambda_1 \geq \lambda_2$  or  $\lambda_2 \leq \lambda_1$ . If  $\lambda_1 \geq \lambda_2$  and  $\lambda_1$  doesn't equal to  $\lambda_2$ , we say that  $\lambda_1$  strictly dominates  $\lambda_2$ , and represent it by  $\lambda_1 > \lambda_2$  or  $\lambda_2 < \lambda_1$ . Two labels are incomparable if none of them can dominate the other. Two rules constrain the authorization under BLP model:

**Rule 1:** (simple security property) Subject  $S$  can read object  $O$  only when  $\lambda(O) \leq \lambda(S)$ .

**Rule 2:** (\* property) Subject  $S$  can write object  $O$  only when  $\lambda(O) \geq \lambda(S)$ .

Rule 1 indicates that low-level users cannot read any high-level information. Rule 2 indicates that high-level users cannot write data to low-level objects. However, when applying BLP policy to data with good structures, such as database, rule 2 have to be modified to maintain the integrity of the data. In multilevel database, writing data is allowed only when the users and objects have the same label. The objects in XML documents also have well-formed structure. So we use strict BLP model for XML documents. The \* property is replaced by the strict \* property:

**Rule 3:** (Strict \* property) Subject  $S$  can write object  $O$  only when  $\lambda(O) = \lambda(S)$ .

### 3 XML Documents Model with Labels

In fine-grained BLP policy on XML documents, every element and attribute is the object and has the label. We first extend the XML document model to include the label information.

#### 3.1 Extended XML Documents Model

**Definition 1:**  $XDOC = (V_e, v_r, V_a, N_s, L_s, subelem, attr, name, label)$ , and

- $V_e$  is a set of all the elements in the document.
- $v_r$  is the root element of the document,  $v_r \in V_e$ .
- $V_a$  is a set of all the attributes in the document.
- $N_s$  is a set of the names of all the nodes in the documents.
- $L_s$  is a set of all the labels in the documents.
- $subelem$  is a many-to-one binary relation.  $subelem \subseteq V_e \times V_e$ . If  $e_1$  and  $e_2$  are the members of  $V_e$ ,  $(e_1, e_2) \in subelem$  shows that  $e_1$  is a sub-element of  $e_2$ .
- $attrs$  is a many-to-one binary relation.  $attrs \subseteq V_a \times V_e$ .  $(a_1, e_1) \in attrs$  shows that  $a_1$  is an attribute of  $e_1$ . Some attributes of an element may be links to other elements using *REFER*,
- $name$  is a one-to-many binary relation.  $name \subseteq N_s \times (V_a \cup V_e)$ .  $(n_1, v_1) \in name$  shows that  $n_1$  is the name of  $v_1$ .
- $label$  is a one-to-many binary relation.  $label \subseteq L_s \times (V_a \cup V_e)$ .  $(l_1, v_1) \in label$  shows that  $l_1$  is the label of  $v_1$ . We also represent it by  $l_1 = \lambda(v_1)$ .

Figure 1 shows an example of XML document, and its schema. *Employees.xml* contains some information of all the employees in a company, and it is valid against an XML schema file named “*employees.xsd*”. Without loss of generality, we will discuss our method based on this example.

#### 3.2 Labels in XML Documents

We will define three kinds of labels in XML documents to manage the security attributes of the objects in this section.

<pre> &lt;employees&gt;   &lt;employee name="Bill"&gt;     &lt;department&gt;Manage     &lt;/department&gt;     &lt;office&gt;No.415&lt;/office&gt;     &lt;phone&gt;8215&lt;/phone&gt;     &lt;salary&gt;15000&lt;/salary&gt;   &lt;/employee&gt;   &lt;employee name="Mary"&gt;     &lt;department&gt;Personnel     &lt;/department&gt;     &lt;office&gt;No.311&lt;/office&gt;     &lt;phone&gt;8327&lt;/phone&gt;     &lt;salary&gt;7000&lt;/salary&gt;   &lt;/employee&gt;   &lt;employee name="John"&gt;     &lt;department&gt;Sales     &lt;/department&gt;     &lt;office&gt;No.306&lt;/office&gt;     &lt;phone&gt;8364&lt;/phone&gt;     &lt;salary&gt;8000&lt;/salary&gt;   &lt;/employee&gt; &lt;/employees&gt; </pre>	<pre> &lt;? xml version="1.0" encoding="UTF-8"?&gt; &lt;xsd:schema   targetNamespace="http://myweb"   xmlns:xsd="http://www.w3.org/2001/XMLSchema"&gt;   &lt;xsd:element name="employees"&gt;     &lt;xsd:complexType&gt;       &lt;xsd:sequence base="employee" maxOccurs="unbounded"&gt;         &lt;xsd:element name="employee"&gt;           &lt;xsd:complexType&gt;             &lt;xsd:sequence&gt;               &lt;xsd:element name="department" type="xsd:string"/&gt;               &lt;xsd:element name="office" type="xsd:string"/&gt;               &lt;xsd:element name="phone" type="xsd:string"/&gt;               &lt;xsd:element name="salary" type="xsd:positiveinteger"/&gt;             &lt;/xsd:sequence&gt;             &lt;xsd:attribute name="name" type="xsd:string"/&gt;           &lt;/xsd:complexType&gt;         &lt;/xsd:element&gt;       &lt;/xsd:sequence&gt;       &lt;xsd:key name="NameKey" selector="xpath=../../employee" field="name"/&gt;     &lt;/xsd:complexType&gt;   &lt;/xsd:element&gt; &lt;/xsd:schema&gt; </pre>
(a) employee.xml	(b) employee.xsd

Fig. 1. Example of XML Document and Its Schema File

### 3.2.1 Assigned Label

The administrator has the privilege to assign labels to the objects explicitly.

**Definition 2:** (Assigned Label) Assigned label of a node is the label assigned to it by the administrator.

The assigned label of an element may dominate, equal to, be dominated by or be incomparable with the one of its sub-elements and attributes. A node can only have one assigned label. If the administrator reassigns a label to a node, the new assigned label will replace the old one.

### 3.2.2 Propagated Label

Some XML environments have too many XML nodes to assign labels explicitly. In those systems only processing valid documents, lots of XML documents may conform to a certain XML schema[18] file. The nodes in these documents may have similar security attributes. For convenience, we can assign labels to the elements of an XML schema that defines the nodes in the XML document instances. Then the label propagates to the instance nodes. We use *d-element* to represent those elements in XML schema that are responsible for defining nodes. When a label is assigned to a *d-element*, the label will propagate to all the elements or attributes defined by the *d-element*. This kind of label is called propagated label in our model.

**Definition 3:** (Propagated Label) Propagated label of a node is the label assigned to the *d-element* that defines the node in the XML schema.

Although an node in XML documents has propagated label, administrator can assign another label to it. Compare to propagated label, assigned label has higher priority



because it is more specifically. However, the value of assigned label is limited by the propagated label. The propagated label of a node defines its lowest label. Rule 4 constraints the relation of the assigned label and the propagated label.

**Rule 4:** The assigned label of a node should dominate its propagated label.

### 3.2.3 Inherited Label

Since the number of the nodes in a XML-based system is very large, some nodes may have no propagated label or assigned label. However, any object should have the security attribute. There are two policies to determine the label of these nodes.

- Use the lowest label in the system,
- Use the same label as its container.

Apparently the second policy is more secure than the first one. If the node is an element but not the root element, the container is its parent element. If the node is an attribute, the container is the element that it belongs to. As a node is considered to be a part of its container, we prefer using the second policy. So in our model, a node may inherit label from its container. We called it “inherited label”.

**Definition 4:** (Inherited Label) Inherited label of a node is the label of its container.

Root element has no any container. If the root element of an XML document has no propagated label or assigned label, its label is the lowest label in the system.

## 3.3 Determining the Labels of the Nodes in XML Documents

Figure 2 gives the algorithm of determining the label of any node in XML documents. When use this algorithm, the worst situation is that we have to get the labels of all the nodes in the path from the root to the determining node, so the algorithm will take  $O(N)$  time, where  $N$  is the max depth of the XML document tree. The inherited label of a node can be acquired using this algorithm repeatedly, while the propagated label and assigned label are stored in some places. If we put propagated labels in XML schema files or store assigned labels in XML documents, we have to change the structures of the schema files and documents. Then our system cannot exchange documents with other applications directly. And we also must modify the structures of existed XML documents when enforcing MAC policy on them. So we decide to store propagated labels and assigned labels into other XML documents – Security Attribute Files(SAF). We will create a SAF to store propagated labels for every XML schema file. All the documents valid against a schema file can share the same SAF. We create a SAF for an XML document only when its nodes have the assigned label. The number of SAFs will be much smaller than the one of the XML documents in a system. And the scale of SAFs is also largely smaller than the XML documents. Figure 3(a) is a SAF for employee.xsd. Figure 3(b) is a SAF for employee.xml. Because of rule 4,  $\lambda_3$  must dominate  $\lambda_1$ . Table 1 gives the labels of some elements in employee.xml determined by the algorithm in figure 2.

Determining the label of the node  $v_j$  in the XML documents:

1. If the node  $v_j$  has an assigned label  $\lambda_j$ , then  $\lambda_j$  is the label of  $v_j$ . Return  $\lambda_j$ .
2. If  $v_j$  has a propagated label  $\lambda_2$ , then  $\lambda_2$  is the label of  $v_j$ . Return  $\lambda_2$ .
3. If  $v_j$  is root element, get the lowest label in the system. Suppose the lowest label is  $\lambda_3$ , then  $\lambda_3$  is the label of  $v_j$ . Return  $\lambda_3$ .
4. If  $v_j$  is not root element, it must be a sub-element or attribute of another element represented by  $v_2$ . Get the label of  $v_2$  using this algorithm. Suppose  $\lambda_4$  is the label of  $v_2$ .
5.  $\lambda_4$  is the label of  $v_j$ . Return  $\lambda_4$ .

**Fig. 2.** Algorithm for Determining Labels of Nodes in XML Documents

```
<SecurityAttributes>
<PropagatedLabel>
  <Object>//xs:element[name="employee"]
</Object>
  <Label> $\lambda_1$ </Label>
</PropagatedLabel>
<PropagatedLabel>
  <Object>//xs:element[name="salary"]
</Object>
  <Label> $\lambda_2$ </Label>
</PropagatedLabel>
</SecurityAttributes>
```

(a) SAF Storing Propagated Labels

```
<SecurityAttributes>
<AssignedLabel>
  <Object>
    /employees/employee[name="Bill"]
  </Object>
  <Label> $\lambda_3$ </Label>
</AssignedLabel>
</SecurityAttributes>
```

(b) SAF Storing Assigned Labels

**Fig. 3.** Examples of SAFs

**Table 1.** Labels of some elements of employee.xml

Object	Label	Source
/employees/employee[name="Mary"]	$\lambda_1$	Propagated Label
/employees/employee[name="Bill"]	$\lambda_3$	Assigned Label
/employees/employee[name="John"]/salary	$\lambda_2$	Propagated Label
/employees/employee[name="Bill"]/salary	$\lambda_2$	Propagated Label
/employees/employee[name="John"]/office	$\lambda_1$	Inherited Label

## 4 Security Views of XML Documents Under MAC Policy

Under MAC policy, some users can only query a part of an XML document because their label cannot dominate the one of all the nodes. The part of XML documents is called "security view" in our model.

### 4.1 Generation of the Security Views

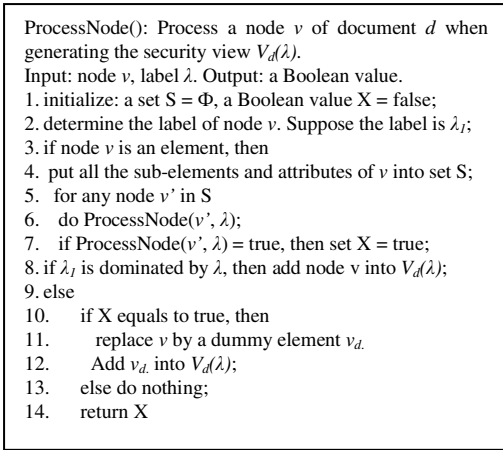
Our security views are generated according to the labels of the nodes in XML documents. For every label in the system, we will generate a security view for an XML documents. If an XML document is represented by  $d$ , the security view of  $d$  on label  $\lambda$  is  $V_d(\lambda)$ . All the users with the label  $\lambda$  share the same security view  $V_d(\lambda)$ .

**Definition 5:**  $V_d(\lambda)$  is the security view of the XML document  $d$  on label  $\lambda$  under MAC policy. It contains all the nodes which label is dominated by  $\lambda$  in  $d$ .

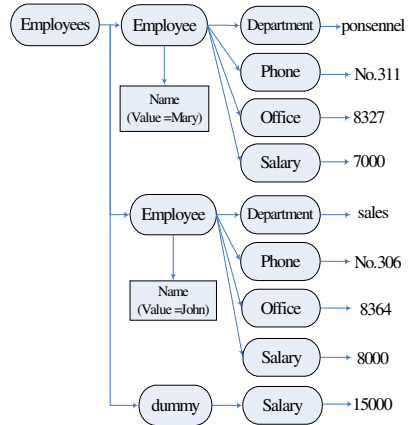
The structure of a security view will be different with the original document because those nodes with higher level label will not appear. However, some elements with higher level label can not be eliminated when it has at least one direct or indirect sub-element or attribute with lower level label. To ensure the semantics of the view, we replace these elements by a “dummy” element. The dummy element has not any information. The details of the function of processing nodes in a document when generating the security views –  $ProcessNode()$  – are presented in figure 4. This function will call itself recursively to process the sub-elements and attributes. If  $r$  is the root element of the document  $d$ ,  $ProcessNode(r, \lambda)$  will process all the nodes in  $d$ . Since every node in a document is processed only once, the algorithm will take the  $O(D)$  time, where  $D$  is the size of the document. Suppose  $\lambda_2 > \lambda_1, \lambda_3 > \lambda_1$  and  $\lambda_3$  is incomparable with  $\lambda_2$ . Figure 5 is the graphical representation of  $V_{employee.xml}(\lambda_2)$ .

### 4.2 Validity of Security View Under MAC Policy

The security views of a valid document should be also valid. However, a view is only a part of the document. So it should be valid to only a part of the schema. The dummy elements will not be considered when we checking the validity of the security view.  $P_s(\lambda)$  is the part of the schema on label  $\lambda$ , and Rule 5 describes the meaning of the validity of security views under MAC policy



**Fig. 4.** Function of Processing Node



**Fig. 5.** Graphical Representation of  $V_{employee.xml}(\lambda_2)$

**Definition 6:**  $P_s(\lambda)$  is the part of the schema file  $s$  on label  $\lambda$ . It does not contain those  $d$ -elements which label assigned by the administrator is not dominated by  $\lambda$ . The sub-elements (are also  $d$ -elements) of those  $d$ -elements are not contained yet although their label may be dominated by  $\lambda$ .

**Rule 5:** If a document  $d$  is valid against a schema  $s$ ,  $V_d(\lambda)$  is always valid against  $P_s(\lambda)$  for any label  $\lambda$  in the system.

To maintain the validity of security views under MAC policy, the labels of the nodes in an XML document have to conform to some extra rules. These rules will constraint the assignment of the labels.

#### 4.2.1 Rules Related to Elements to Maintain the Validity

The number of sub-elements in an element may be constrained by the minimum and maximum occurrences. Rule 6 limits the number of the elements that have minimum occurrence attribute in their definitions.

**Rule 6:** If  $n_1$  is the name of some sub-elements in an element named  $n_2$ , and the number of  $n_1$  is constrained by the minimum occurrences  $min$  in the definition of  $n_2$ , and if  $d_1$  is any document that is valid against the schema file, in every  $n_2$  of  $d_1$  there are at least  $min$  sub-elements  $n_1$  which label equals to the propagating label of  $n_1$  or be dominated by the label of  $n_2$  if  $n_1$  has not propagated label.

With the rule 6, if  $P_s(\lambda)$  contains the  $d$ -element of the sub-elements  $n_1$  in the element  $n_2$ , any element  $n_2$  in  $V_d(\lambda)$  will have more sub-elements  $n_1$  than the value of the minimum occurrence attribute of the sub-element  $n_1$ . Another kind of constraints on the element definitions is maximum occurrence. But the valid security view may violate this kind of constraints under MAC policy. In some situations we use rule 7 to limit the number of elements that have maximum occurrence constraints.

**Rule 7:** If  $n_1$  is the name of some sub-elements in an element named  $n_2$ , and the number of sub-element  $n_1$  is constrained by the maximum occurrences  $max$  in the definition. If  $d_1$  is any document that is valid against the schema file, in every element  $n_2$  of  $d_1$  there are at most  $max$  sub-elements  $n_1$  that have the same labels.

Rule 7 don't constraint the total number of the sub-elements when these sub-elements have different labels. As the low level security views have not the high level objects, the users may insert a sub-element into the security views without knowing the total number of the sub-elements. System cannot reject the insert operation, because a covert channel based the rejection may be utilized. So the rule 7 only constrains the number of the sub-elements with the same label. However, if the bandwidth of the covert channel based on the rejection is too small to be harmful, the rule 7 may not be used. When the administrator assign or reassign labels to the objects, the results cannot violate the rule 6 and rule 7.

#### 4.2.2 Rules Related to Attributes to Maintain Validity

Two kinds of attributes can be defined in an XML schema – optional and required. Optional attributes of an element may appear or not appear in the instances of the element. But required attributes of an element must appear in any instance of the element. To maintain the validity of security views under MAC policy, system should guarantee that the required attribute appears in a security view  $V_d(\lambda)$  if  $P_s(\lambda)$  has the definition of the attribute.

**Rule 8:** If an element named  $n_1$  has a required attribute named  $n_2$  in the definition, any instance of  $n_1$  should have an attribute  $n_2$  which label equals to the propagated label of  $n_2$  or be dominated by the label of the instance if  $n_2$  has not propagated label.

## 5 Access Control on XML Documents Under MAC Policy

All the operations on XML documents either are browse type or update type. Update operations on XML documents can be divided into many subtypes because of the complexity structure of XML documents [2, 8]. To make our paper clear, we don't discuss those operations that will change the structure of the schema in this paper. In fact, our discussion will be easy to be extended to include those operations. The operations we discuss fall into four kinds: query, insert, delete and modify. Table 2 is the simple descriptions for these operations. We suppose that the users have been authorized to do these operations. In fact, if the users' label cannot dominate all the nodes' labels in the documents, their operations will be redirected to a security view. The results of the update operations can not violate the validity of the security view under MAC policy.

### 5.1 Query Operations on Elements and Attributes

The simple security property of BLP model limits that the users can read an object only when their label dominates the one of the object. If a user sends query operations on a document, the system will return the query results from the a security view generated according to the document and the user's label. Because the security view does not include the nodes which label cannot be dominated by the user's label, query results will not contain these nodes. For example, suppose user  $U$  with label  $\lambda_I$  sends a XPath[17] query operation “//employee/\*” on the document employee.xml. The operation will be redirected to  $V_{\text{employee.xml}}(\lambda_I)$ . The contents of two “employee” elements will be returned to  $U$ . The “employee” element which name equals to “Bill” will not returned because it is not in  $V_{\text{employee.xml}}(\lambda_I)$ .

**Table 2.** Operations on XML Documents

Operation	Description
Query	Query the content of an element or the value of an attribute
Insert	Insert a new sub-element or attribute into an element.
Delete	Delete a sub-element or attribute from an element
Modify	Modify the content of an element or the value of an attribute

### 5.2 Insert Operation on Elements

According to the rule 3, the data created by a user will have the same label as the one of the user. Using above example, suppose user  $U$  sends a request to insert a new “employee” element  $v_I$  into employee.xml.  $v_I$  will be inserted into  $V_{\text{employee.xml}}(\lambda_I)$  and be inserted into employee.xml at last. The label of  $v_I$  should be  $\lambda_I$  which is the same as  $U$ 's label. System will assign the operator's label to  $v_I$  automatically. The sub-elements and attributes of  $v_I$  inherited label from  $v_I$ . However,  $U$  cannot insert “salary” sub-element into  $v_I$  for the reason that  $P_{\text{employee.xsd}}(\lambda_I)$  dose not contain the  $d$ -element of “salary”. To keep the validity of employee.xml for the users with higher label, we will add a “salary” element with empty or default value into  $v_I$  when inserting  $v_I$  into employee.xml.

### 5.3 Delete Operation on Elements

Deleting a sub-element or attribute from an element is also a kind of writing operation. According to rule 3, one can only delete those sub-elements or attributes which label equals to one's label. Suppose  $U$  is authorized to delete sub-elements of "employees" element. Then  $U$  can delete the two "employee" elements in  $V_{\text{employee.xml}}(\lambda_U)$ . But the two elements have sub-elements "salary" that are invisible to  $U$ . System cannot refuse the operation directly because users may acquire high-level information through covert channels based on it. Two policies can be used to solve the problem. One policy is deleting all the sub-elements and attributes no matter whether they have higher label. The other policy is lifting up the label of the deleted elements for those users with higher label.

### 5.4 Modify Operation on Elements or Attributes

Modify operations will change the content of an element or the value of an attribute. As long as being authorized, users can modify any nodes in a security view. But the new data should have the same label as the operators. If the target has the same label as the operators, we just modify the content or the value. But if the target has the lower label than the operators, they cannot change content or value directly. There are two ways for a user to modify the objects that has lower label. The first way is that the user logs on system using the same label as the objects. In many flexible realization of BLP model, a user can log on the system with any label dominated by the users' assigned label. The second way is conserving the original node and inserting a new node into the XML document. The new node has the same label as the users.

### 5.5 Polyinstantiation in XML Document Under MAC Policy

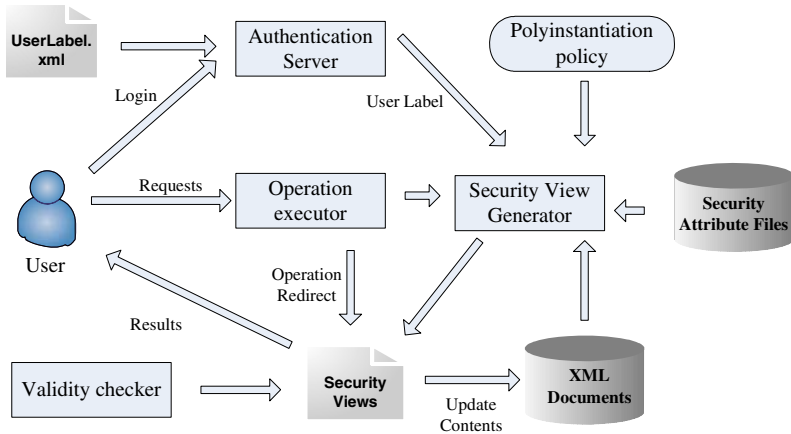
Polyinstantiation may happen after some operations in multilevel database [14]. For example, two tuples with the same primary key will exist in a table when a high-level user updates non-key fields of a low-level tuple. Polyinstantiation will also happen in XML documents under MAC policy. Similar with the tuple, an element also has the primary key if XML schema defines a key for the element. The value of the key is unique for every instance of the element in an XML document. For example, the "name" attribute of element "employee" is the primary key of every "employee" in `employee.xsd`. Polyinstantiation occurs if two "employee" elements with different labels have same value in that attribute. Two kinds of operations may cause polyinstantiation of the elements. First kind is inserting an element without knowing that a high-level element has the same key. Another kind is modifying non-key contents of a low-level element using the second way discussed in section 6.4. Polyinstantiation in XML documents under MAC policy also occurs if an element has two attributes with same name but different label. Similarly with the elements, two kinds of operations will cause polyinstantiation of attributes. One kind is creating a new optional attribute without knowing that a high-level attribute with same name exists. Another is modifying the value of a low-level attribute using the second way.

If there is more than one of the polyinstantiation nodes which label is dominated by the user's label, how to process those nodes is a problem when generating the security views. In some strict systems, we will select only one of the polyinstantiation nodes to

add into the views. The node with the same label as the user has the most priority to be selected. If none of these nodes has the same label as the users, system will select one. The administrator could define a total order relation among the labels in the system. Then the system is able to select one by comparing their labels. In other systems, they will add all the proper polyinstantiation nodes into the security views. These nodes will be marked with their labels. Users can query any one of these nodes.

## 6 Architecture of Enforcing MAC Policy on XML Documents

Figure 6 is the architecture of enforcing MAC policy on XML documents. As discussed above in section 4.3, labels of the nodes will be stored in the SAFs. In our architecture, the SAFs are XML documents. The file used to store the users' label - UserLabel.xml is also an XML document. So the XML mechanisms can be used to process these files. These files can only be accessed or updated by the administrator or kernel modules of the system.



**Fig. 6.** Architecture of Enforcing MAC Policy on XML documents

Before sending any request, a user has to login the system through the authentication server. Then the server queries the user's label from UserLabel.xml. The user's label is one of the inputs of security view generator. When the operation executor receives the user's request, it will ask the security view generator to provide a proper security view. Then the user's requests are redirected to the security view. If the user's request contains any update operation, validity checker will check whether the views and documents are still validity after the operation. If not, the operation will be refused. The security view generator is the critical module to enforce MAC policy in our architecture. The inputs of security view generator include: XML documents, SAFs, users' labels and polyinstatiation policy. To improve the efficiency, the security view generator will save the security view for future use. Only when the original documents or SAFs have been updated, the security views would be generated once again.

## 7 Related Work

XML access control has been discussed in many papers. XACML [11] is a specification of OASIS to express and deploy access control policy based on XML. E. Dimiani and E. Bertino etc discussed access control for XML documents based on DTD technology [4, 5, 6]. A fine-grained access control model is proposed in [7]. Access privileges on elements are defined in this model. [2] discusses an XML access control model and propose a technique that supports update operations. The concept of “Restrict views” to implement security in XML documents is proposed by [1]. All the models above are based on DAC policies. RBAC for XML has also been discussed. DTD files to implement RBAC model was given in [12]. M.Hitchens etc [9] presented a RBAC model for XML document stores, use their own language to describe roles and permissions. Practical concepts that can be employed in an enterprise environment for managing security policies using XML were described in [10]. They discussed how to implement RBAC using Java and XML technologies. An extended RBAC model for XML security based on XML schema components was presented in [19]. Although RBAC can be configured to enforce DAC and MAC policies[13], the mechanisms and technologies of enforcing MAC on XML documents need to be discussed specifically.

This paper discussed the details and architecture of how to enforce MAC policy on XML documents. Determining the labels of the nodes and processing operation on XML documents under MAC policy are presented. Our method is based on the XML security views under MAC policy. [15] proposed the concept of the security view for XML documents to implement security XML querying. Their XML security views are generated according to the authorization of the users. Our security views are different with them. We generated the XML security views according to the labels of the nodes in XML documents. Our function of generating security views is very easy. The validity of the security views is also discussed.

## 8 Conclusion

XML Documents in military and highly secure systems need to be protected by MAC policy. This paper addresses how to enforce MAC policy on XML documents. An extended XML document model that contains the labels is presented. Three kinds of labels will decide the security attributes of the nodes in XML documents. Assigned labels are those labels assigned by the administrator specifically. The nodes of a valid document may have propagated labels that are assigned to the definitions of the nodes. And a node can inherit the label from its container. The algorithm of determining the label of the nodes in XML documents is also given. The security view under MAC policy is defined as a part of the XML document. The security view on label  $\lambda$  contains only the nodes which label is dominated by  $\lambda$ . We have also discussed the validity of a security view against a part of the schema. Once generated the security views, the operations of a user will be redirected to a view according to the user’s label. We have discussed four kinds of operations on XML documents under MAC policy. Polyinstantiation is inevitable when enforcing MAC policy. We described the polyinstantiation of elements or attributes in XML documents after some operations. At



last, we have presented our architecture of enforcing MAC policy on XML documents. XML technologies are used in most parts of the architecture to make the implementation more easily.

## References

1. Abhilash Gummadi, Jong P. Yoon, Biren Shah, Vijay Raghavan: A Bitmap-based Access Control for Restricted Views of XML Documents, In Proc. of the 2003 ACM workshop on XML security, Fairfax, Virginia, USA, October 2003, 60-68.
2. Chung-Hwan Lim, Seog Park, Sang H. Son: Access Control of XML Documents Considering Update Operations, In Proc. of the 2003 ACM workshop on XML security, Fairfax, Virginia, USA, October, 2003, 49-59.
3. D. Elliott Bell, Leonard J. LaPadula: Secure Computer Systems: Unified Exposition and Multics Interpretation” MITRE Corporation, Bedford, MA, USA, ESD-TR-75-306, NTIS #AD-A023588, March 1976.
4. Elisa Bertino, Silvana Castano, Elena Ferrari, Marco Mesiti: Specifying and Enforcing Access Control Policies for XML Document Sources. *World Wide Web* 3(3): 139-151, 2000.
5. E. Beritino, S.Castano, E.Ferrai: Securing XML documents with Author-x, *IEEE Internet Computing*, May/June, 2001, 21-31.
6. E. Damiani, S.D.C.Vimercati, S.Paraboschi, and P.Samarati: Design and Implementation of Access Control Processor for XML Documents. *Computer Network*, Volume 33, Issue 1-6, June 2000, 59-75.
7. Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati: A Fine-grained Access Control System for XML Documents. *ACM TISSEC* 5.2, May 2002, 169-202.
8. I. Tatarinov, Z. Ives, A. Halevy, D. Weld: Updating XML. In Proc. of the 2001 ACM SIGMOD International Conference on Management of Data, California, USA, May 2001, 413-424.
9. M.Hitchens, V.Varadharajan: RBAC for XML Document Stores. In Proc. of International Conference on Information and Communications Security, Xian, China, November, 2001, 131-143.
10. Nathan N. Vuong, Geoffrey Smith, Yi Deng: Managing Security Policies in a Distributed Environment Using eXtensible Markup Language (XML). In Proc. of Eighth Annual Workshop on Selected Areas of Cryptography, Toronto, Canada, August, 2001, 405-411.
11. OASIS standard: eXtensible Access Control Markup Language (XACML) Version 1.0. <http://www.oasis-open.org/committees/xacml/repository/oasis-xacml-1.0.pdf>, 18 February 2003.
12. R. Chandramouli: Application of XML Tools for Enterprise-Wide RBAC Implementation Tasks. In Proc. of 5th ACM workshop on Role-based Access Control, Berlin, Germany, July 26-27, 2000, 11-18.
13. S. Osborn, R. Sandhu and Q. Munawer: Configuring Role -Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM TISSEC*, 3(2): 85-106, May 2000.
14. Sushil Jajodia, Ravi S. Sandhu, and Barbara T. Blaustein: Solutions to the Polyinstantiation Problem. *Information Security: An Integrated Collection of Essays*, Essay 19. IEEE Computer Society Press, Los Alamitos, 1995.

15. Wenfei Fan, Chee-Yong Chan, Minos Garofalakis: Secure XML querying with security views, In Proceedings of the 2004 ACM SIGMOD internal conference on the management of data, Paris, France, June, 2004, 587-598.
16. World Wide Web Consortium (W3C): Extensible Markup Language (XML) 1.0, <http://www.w3.org/TR/REC-xml>, October, 2000.
17. World Wide Web Consortium (W3C): XML Path Language (XPath), <http://www.w3.org/TR/xpath20>, August, 2002.
18. World Wide Web Consortium (W3C): XML Schema Part 0: Primer, <http://www.w3.org/TR/xmlschema-0>, May, 2001.
19. Xinwen Zhang, Jaehong Park and Ravi Sandhu: Schema based XML Security: RBAC Approach, Technical Report, IFIP WG 11.3, 2003.

# Network Access Control for Mobile Ad-Hoc Networks\*

Pan Wang, Peng Ning, and Douglas S. Reeves

North Carolina State University,  
Raleigh NC 27695, USA  
{pwang3, pning, reeves}@ncsu.edu

**Abstract.** In this paper, we propose to enforce network access control in Mobile Ad Hoc Networks (MANETs) using cryptographic techniques. In the proposed approach, packets are authenticated by means of a network-wide symmetric (session) key. Because nodes are mobile and communication paths may change rapidly, timely distribution of new session keys is challenging (particularly if keys change frequently). Nodes wishing to communicate may therefore hold different session keys, which must somehow be synchronized. We present a fully distributed key synchronization method based on stateless group key distribution, and localized packet retransmission. If nodes A and B wish to communicate securely over a path  $P$ , all nodes on this path must synchronize keys with their immediately adjacent neighbors in the path. Any node which is unable to synchronize keys will not be allowed to forward packets. Simulations and a functioning prototype demonstrate the proposed system is practical and effective.

## 1 Introduction

A MANET is an autonomous system formed by a set of wireless mobile nodes that generally operate with low battery power and with limited bandwidth. Currently, most MANETs do not have any network access control mechanism. The attackers may thus easily gain access to the network, and launch various attacks. For instance, an attacker may inject a large number of “bogus” or spurious packets into the network, simply to consume network bandwidth and the battery power of mobile nodes.

Firewalls have been typically been used to enforce network access control, using network topology and service information. They are particularly valuable for guarding against resource consumption. However, the dynamic nature and open (wireless) medium of MANETs make it difficult to directly apply techniques such as firewalls.

In this paper, we propose to enforce network access control in MANETs using cryptographic techniques. That is, a network-wide access control (secret session) key, which is only known by approved nodes, is employed to authenticate all the packets transmitted in the network. Each node asked to forward a packet inspects the packet to see if it has been authenticated with an access control key as recent as the one it (the forwarding node) holds. Only packets meeting this test are forwarded by the MANET.

A critical challenge is how to manage such access control keys in a MANET. Due to the need to remove compromised nodes, this access control key may have to be updated

---

\* This work is supported by the US National Science Foundation (NSF) under grants CCR-0207297 and CAREER-0447761.

frequently. The dynamic network topology and frequent communication failures make it difficult to achieve a globally-synchronized key after each key update. To the best of our knowledge, no existing key distribution schemes can ensure such key synchronization, *even if* the key manager is assumed to be always online.

To address this issue, we propose a distributed key synchronization method that makes use of *stateless group key distribution* schemes (e.g., [5, 11, 13]). These schemes have the nice property that a legitimate node can get the updated group key from a received key update message, even if that node has missed several previous rounds of key updates. The proposed key synchronization method guarantees that whenever legitimate nodes communicate with each other, they will synchronize their access control keys and agree on the latest one needed to authenticate and verify data packets.

Another challenging problem is how a node deals with received packets that cannot be verified immediately, i.e., that are authenticated with keys different than the one it holds. This can happen if a new access control key has not been fully distributed throughout the network. To address this problem, we present a packet retransmission scheme. By requiring the sender to locally retransmit the data packet, this scheme limits the propagation of unverified packets to a single hop, and allows the involved nodes to synchronize their access control keys and establish or continue communication.

The rest of the paper is organized as follows. Section 2 introduces the underlying models and cryptographic tools used in this paper. Section 3 discusses the proposed system and the critical techniques for synchronizing session keys. Section 4 describes the preliminary results of implementation and system testing. Section 5 discusses related work. Section 6 concludes this paper and points out some future research directions.

## 2 Underlying Models and Cryptographic Tools

### 2.1 Network Model

We notice that without a logically centralized authority, a faulty entity can behave arbitrarily and thus defeat an access control system. Therefore, we assume all legitimate nodes come from one domain and their accesses to MANET are controlled by a *key manager* that is not required to be always online. We argue that this assumption is reasonable, if we want to enforce access control in MANETs.

All nodes register at the key manager offline before joining the network. Each of them is pre-configured with a unique (ID), ID certificate, current network access control key, and a set of personal secret keys used for stateless key distribution purposes. We also assume neighboring nodes know each other's ID by verifying the ID certificate.

Whenever a new network access control key needs to be distributed, e.g., after a compromised node is detected, the key manager broadcasts a key update message to the network, or gives the key update message to a newly joined node. Note that the methods for detecting compromised nodes are not in the scope of this paper.

### 2.2 Attack Model

The proposed network access control system relies on cryptographic techniques to control the nodes' access to a MANET. An attacker with unbounded computing capability

is capable of compromising any practical security protocol. We thus assume that the attackers have bounded computing capability. They cannot break the adopted stateless group key distribution scheme.

An attacker may wish to inject packets into the network with the goal of depleting the resources of nodes relaying the packets. Even though neighboring nodes will not forward bogus packets injected by an attacker, they will have to spend some resources on verifying these packets. Such a local resource consumption attack is possible and cannot be prevented by the proposed system.

### 2.3 Stateless Group Key Distribution

Based on the interdependency of key update messages, group key distribution schemes can be classified into two categories, stateful [10, 14, 15] and stateless [5, 11, 13]. In the stateless group key distribution schemes, each user is preassigned a unique ID and some personal secret keys that never change during the lifetime of the group. To revoke a user or to update the group key, the key manager encrypts a new session key separately, using a set of secret keys only known to the non-revoked users. The manager creates a key update message consisting of the resulting ciphertexts and some auxiliary information (e.g., the IDs of the encryption keys), and then broadcasts this message to the network. After receiving a key update message, a non-revoked node uses its personal secret key (or a key derived from its personal secret key) to decrypt a certain part of the message (indicated by the user's ID), and from this obtains the new session key. As a result, the stateless key distribution scheme has two nice properties. First, a legitimate user can get the update group key as long as the user has the corresponding key update message, *even if* the user is offline for a while, or misses several previous rounds of key updates. Second, a legitimate node can calculate the ID list of all revoked nodes from a received key update message.

### 2.4 One-Way Key Chain

A one-way key chain is a chain of keys generated through repeatedly applying a one-way hash function  $H$  on a random number (key seed). For instance,  $k_{n-1} = H(k_n), \dots, k_0 = H(k_1)$ . The property of one-way means, given a latest released key  $k_i$  from a one-way key chain, it is computationally infeasible for an adversary to find any unreleased key  $k_j$  such that  $H^{j-i}(k_j)$  equals  $k_i$ . However, it allows a receiver to easily verify that a later key  $k_x$  really belongs to the key chain by checking that  $H^{x-i}(k_x)$  equals  $k_i$ .

## 3 Network Access Control for MANET

We propose to employ a symmetric key based network access control system to filter out the bogus packets from a MANET. That is, a key manager (not required to be always online) controls the nodes' accesses to the MANET by selectively distributing a common network access control key. Only authorized nodes which have not been revoked by the key manager can get such a common key.

Each node uses this network access control key to authenticate every outgoing packet, i.e., to insert a *Packet Authentication Header* (PAH). A possible way of adding the PAH, which is used in our implementation, is illustrated by Appendix A. Each node also uses this key to verify the PAH of an incoming packet, either intended for this node or transiting through it. They will immediately drop any packet not properly authenticated. As a result, a non-authorized node that is either unknown to or revoked by the key manager is prevented from injecting packets into the MANET.

Clearly, the network access control key may have to be updated due to the revocation of compromised nodes. Since each key update (rekeying) event defines a key session, the access control keys are also referred to as session keys. In this paper, we use network access control key and session key interchangeably, and we assume a legitimate node only uses the most recent session key it has received.

To make the proposed symmetric key based network access control system practical, two critical issues need to be solved.

1. The proposed system requires all legitimate nodes agree on the same key at the time they communicate. However, no existing group key distribution scheme can guarantee that all nodes receive the latest session key after a key update. Maintaining a synchronized session key therefore becomes a non-trivial task, given the dynamic nature of MANETs and the possibility of intermittent communication failures.
2. Two legitimate nodes cannot communicate in the proposed system, if a more recent key update message only reaches one of them, unless they can agree on one specific session key. Thus, another challenging problem is how a node handles the received packets authenticated with different session keys, to ensure secure establishment/continuation of communication with corresponding senders.

In the following subsections, we present solutions to these two issues. We first introduce a key synchronization method, which exploits the stateless feature of stateless group key distribution schemes. This method assists in the distribution of the latest session key to all legitimate nodes. Next, we introduce a packet retransmission scheme when two communicating nodes have different session keys. This scheme allows the involved nodes to synchronize their keys, and ensures the data is delivered and correctly authenticated. Note that, we use the *source* and the *destination* to denote the two end-hosts along a communication path, i.e., the node where a packet is produced and the node where the packet ends up, and use the *sender* and the *receiver* to denote the two communicating nodes adjacent to one another.

### 3.1 Synchronization of Network Access Control Keys

In the proposed network access control system, ideally, all legitimate nodes should use the same (latest) session key at any time. However, due to communication failures, the limited range of wireless transmission, and changes in the network topology, a key update message may fail to propagate across the entire MANET. As a result, two legitimate nodes may simultaneously hold different session keys. We refer to such a scenario as *key un-synchronization*. Key un-synchronization is a fatal threat to the proposed network access control system. It prevents key-unsynchronized nodes from establishing normal communication.

**Definition 1.** Let  $U$  denote a set of legitimate nodes, and  $K_U^{(t)}$  denote the set of session keys used by the nodes in  $U$  at time  $t$ .  $U$  is *key-synchronized* at time  $t$ , if  $|K_U^{(t)}| = 1$ . Otherwise, we say  $U$  is *key-unsynchronized*.

Several possible methods could be employed to remedy such key un-synchronization based on assistance from the key manager. For instance, nodes that become aware they have not received the newest session key could directly contact the key manager to get the key, or the key manager could periodically rebroadcast the key update messages. However, such methods are overly-dependent on the key manager. The key manager could be unreachable by a large fraction of nodes due to temporary partitioning of the network, *even if* it is always online. In such a case, nodes that can physically reach each other by some path will not be able to synchronize their session keys.

In the following, we present a distributed key synchronization method. The detailed algorithm of the proposed method will be presented in the next subsection, due to the interaction between key synchronization and data packet transmission.

**Proposed Method.** The proposed key synchronization method depends upon recent advances in stateless group key distribution [5, 11, 13]. It helps distribute the latest session key to all legitimate nodes, and it guarantees that whenever two legitimate nodes communicate with each other, they will synchronize their session keys and agree on the most recent one to use.

In the proposed key synchronization method, each node buffers (without modification) the key update message it most recently receives. It directly transmit this buffered message to other nodes that may be using an older session key. The corresponding nodes can then extract the new session key from the received message, since a stateless key distribution scheme allows a legitimate user to get the updated group key as long as the user has the corresponding key update message, *even if* the user has been off-line for a while, or missed several previous rounds of key updates. In this way, two key-unsynchronized nodes that want to communicate can synchronize their session keys without relying upon the key manager.

We notice that an attacker may send ‘bogus’ key update messages. This could lead to a resource consumption attack, if the verification of a key update message involves expensive operations (e.g., asymmetric cryptographic operations). To mitigate this attack, all session keys in the proposed system are generated from a one-way key chain. It sequentially uses these session keys for network access control, and pre-distributes  $k_0$  or the current session key  $k_i$  to a legitimate node as the commitment of the key chain. As discussed earlier, a node thus can verify the authenticity of a new session key (or a new key update message) with a limited number of hash operations that are much cheaper than an asymmetric cryptographic operation.

In order to accelerate the convergence of the network access keys in a MANET, we suggest that each node periodically broadcasts a *beacon message*. This message can be either the node’s buffered key update message, or a special, authenticated packet. From a received beacon message, a local receiver can detect whether it is key-synchronized with the sender. If not, the receiver can initiate a procedure of key synchronization. Due to page limitations, we skip the details of employing beacon messages.

### 3.2 Packet Retransmission in Case of Key Un-synchronization

As was mentioned earlier, in the case of key un-synchronization, a legitimate node may receive some packets authenticated with a different session key than the one it holds. For convenience, we refer to such received packets as *unverified packets*, since they cannot be immediately verified by the receiver. Simply accepting or forwarding an unverified packet is not acceptable. Otherwise, an attacker can easily defeat the proposed network access control system by using a spurious PAH.

There are two important design factors: (a) how a receiver synchronizes the session key with the corresponding sender, and (b) whether a receiver buffers an unverified packet. A legitimate node has four major options to avoid key un-synchronization or to handle an unverified packet. These options are:

1. Synchronizes the keys with all nodes along the communication path first, to avoid the key un-synchronization in future data transmission.
2. Buffers the unverified packets first, then synchronizes the session key with the sender, and finally verifies these buffered packets.
3. Simply drops the unverified packets in case of key un-synchronization.
4. Drops the unverified packets and asks the sender to retransmit the packets, while a key synchronization is triggered during the retransmission.

The first three options each have serious drawbacks. The first option cannot exclude the possibility that a transmission cannot complete because of frequent key updates. The second option exposes legitimate nodes to memory consumption attacks. That is, the victim nodes are enforced to buffer the bogus packets authenticated with a “newer” session key, while waiting to receive the necessary session key update message. The third option prevents two legitimate nodes from establishing or continuing communication when they are key-unsynchronized.

In the proposed network access control system, we propose to use the fourth option. It assists the communicating nodes to reestablish communication in the case they become key-unsynchronized, constrains the propagation of any unverified packet to a single hop, and avoids the memory consumption attack since the receiver is not required to buffer any unverified packets.

**Proposed Scheme.** In this subsection, we present the details of the proposed packet retransmission and key synchronization algorithm. We start with the pseudo-code for unicast data packets, shown in Figure 1.

In the packet receiving part, if the sender and the receiver are key-synchronized, i.e., the session ID in the received packet is equal to the receiver’s, the latter verifies the PAH immediately. Otherwise, the receiver needs to synchronize the session key with the sender and requests a packet retransmission.

The receiver sends a retransmission request to the sender, if the sender has a more recent session key. After receiving this retransmission request, the sender retransmits the data packet with its buffered key update message. The receiver then extracts the new session key from the attached key update message, and verifies the retransmitted packet. Conversely, the receiver sends a retransmission request to the sender and attaches its own buffered key update message. The sender extracts the new session key



from the attached key update message, authenticates the data packet with this key, and retransmits the data packet. The receiver can then verify the retransmitted packet. Thus, any unverified unicast packet is prevented from propagating beyond a single hop.

---

*Locally broadcast a beacon message every  $t$  time unites*

*When sending a packet  $P$*   
*insert a PAH to  $P$*

*When receiving a packet  $P$  from sender  $X$*   
*CASE 1:  $P$  is a data packet*  
*if  $P$ 's session ID > node's current one*  
*drop  $P$ , send a retransmission request to  $X$*   
*along with a key synchronization request*  
*else if  $P$ 's session ID < node's current one*  
*if  $X$  is in the list of revoked nodes*  
*do nothing*  
*else send a retransmission request to  $X$*   
*along with the buffered key update message*  
*else verify the PAH in  $P$*   
*if PAH is correct*  
*accept  $P$  and send ACK to  $X$*   
*else drop  $P$*

*CASE 2:  $P$  is a key update message*  
*if  $P$  is authenticated for a new session key*  
*buffer  $P$ , extract new key, forward  $P$ ,*  
*and calculate the list of revoked nodes*  
*else drop  $P$*

*CASE 3:  $P$  is a retransmission request*  
*if a new key update message is attached*  
*extract the new session key and verify PAH in  $P$*   
*if verification proves correct*  
*send ACK to  $X$*   
*else drop  $p$*   
*else retransmit the data packet*  
*along with the buffered key update message*

*CASE 4:  $P$  is an ACK*  
*remove the acked packet from the buffer*

---

**Fig. 1.** Pseudo-code for Key Synchronization and Packet Retransmission

A solution to this problem would require an additional check. In essence, each receiving node must check if the source of a packet has been revoked according to the session key held by the receiver. If so, the packet is dropped and communication fails. Otherwise, the receiver synchronizes the session key with the sender and forwards the packets to the next node in the path. A configuration parameter can be set to implement this more stringent requirement.

The proposed algorithm uses an ACK mechanism to ensure a data packet is received correctly. Thus, it requires the sender to retain a packet until it is ACKed. If the received packet is verified, the receiver replies with an authenticated ACK. The sender is then free to purge the acknowledged packet from its sending buffer.

The proposed algorithm may allow some packets from a revoked node to be successfully delivered and verified, after the node has been revoked, in the following way. Assume there are 3 nodes  $S$ ,  $X$ , and  $D$ , all synchronized with the same session key, and suppose  $S$  sends a packet to  $D$  through  $X$ .  $X$  will verify the packet and forward it to  $D$ , but in the meantime  $D$  may receive a new session key update message (which revokes  $S$  from the group). When  $X$  forwards the packet to  $D$ , the packet will fail to be verified.  $D$  will request retransmission and send the new session key update message to  $X$ .  $X$  will update its session key, reauthenticate the packet, and retransmit it to  $D$ , which will verify the packet and accept it.

It is worth noting that the proposed algorithm must be modified slightly in the case of a *broadcast* data packet. In this case, we assume ACKs are not used, in order to avoid the ACK implosion problem (i.e., network bandwidth exhaustion from transmitting too many ACKs triggered by one broadcast packet). Accordingly, a data packet that is broadcast is not stored by forwarding nodes. A receiver that requests retransmission therefore has to send its retransmission request to the source of the broadcast packet. The source can then perform the normal processing described above, and rebroadcast the packet.

### 3.3 Analysis

**Correctness.** In the proposed system, a node can tell whether it is key-synchronized with the sender by examining the PAH in the received packet. If the sender and the receiver are key-unsynchronized, the receiver will employ the proposed key synchronization algorithm to synchronize their session keys. A more recently buffered key update message is therefore exchanged between these two nodes. Meanwhile, any improperly authenticated key update message will be detected and filtered out since a node can easily verify the authenticity of a key update message based on its current session key. The corresponding node which holds the old key will extract the new session key from the received key update message, and as a result, these two nodes converge on the newer of their two keys.

Suppose source node  $S$  communicates with destination node  $D$  via  $n$  intermediate nodes,  $I_1 \dots I_n$ . Assume all these  $n + 2$  nodes are legitimate (non-revoked), and they use  $v$  session keys,  $v \leq n + 2$ , where  $k_v$  is the latest session key. For convenience, we assume the time interval between two consecutive key updates is at least as long as the propagation delay (including time for key synchronization and retransmission) between nodes  $S$  and  $D$ . Otherwise, the nodes in the path may never be able to synchronize their keys, if new key update messages continually arrive during the delivery of packets. After a node on the path has been reached which holds  $k_v$ , all nodes on the path after that must synchronize on this most recent key, until the destination is finally reached.

**Security.** The proposed system employs a symmetric key to prevent non-authenticated packets from propagating beyond a single hop, and thus enforces the network access control. However, an attacker could launch some attacks against the proposed system.

A malicious node may cause unnecessary communication by broadcasting a forged beacon message, transmitting a key synchronization request, or sending data packets with spurious PAH. The legitimate nodes will respond by either sending a key synchronization request or transmitting their buffered key update messages, wasting both network bandwidth and battery power. It is not possible to completely prevent such attacks in the proposed system. However, a malicious node can only trigger the legitimate nodes to send a key synchronization request, which generally is much smaller than a buffered key update message, by declaring a newer session key has been used. So the effects of such resource consumption attacks are localized. We speculate that identifying such attacks and the source of the attacks is easier as a result.

An inside attacker, i.e., a compromised but undetected and therefore not revoked node, may refuse to forward information. This may cause a logical (key) partition of

the MANET. The proposed system cannot prevent such a logical partition caused by an inside attacker. However, it can be avoided if the MANET is highly-connected. Otherwise, the issue of nodes which refuse to forward packets is known as a “selfish node” problem for MANETs, for which several solutions have been proposed (e.g. [9]).

The proposed system cannot guarantee a global key synchronization in the case of network partitioning. Therefore, a newly revoked node may enter a subnetwork that still uses an old session key and cause damage. This is a limitation of the proposed scheme, and no method based on centralized revocation can overcome such a limitation. However, we argue that the mobility of nodes accelerates the convergence of the session key of the whole MANET. A revoked node will eventually be excluded in the proposed system, if a node using the latest session key joins that subnetwork.

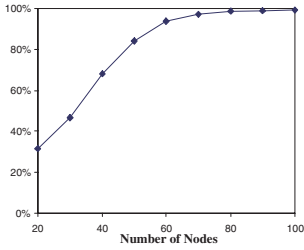
Since the total number of sessions is predetermined by the length of the one-way key chain, an attacker may attempt to deplete the sessions (and thus the lifetime of the network) by frequently joining and leaving, requiring the key manager to frequently update the session key. To mitigate this attack, the key manager can generate a long key chain. Coppersmith and Jakobsson [4] already proposed a scheme to improve the performance of the one-way key chain, which requires only  $O(\log(N))$  storage and  $O(\log(N))$  computation to access an element, where  $N$  is the total number of keys. Furthermore, since a node needs to register at the (centralized) key manager offline to gain network access, repeated registration can be easily identified.

**Overhead.** The computation overhead comes from the operations of packet authentication and verification. A source node needs to calculate a PAH for every outgoing packet, and a receiver needs to verify this PAH before accepting or forwarding the packet. Since symmetric cryptography is used, calculating and verifying a PAH will be very fast and cheap in the proposed system. Such routine operations will not cause a heavy burden.

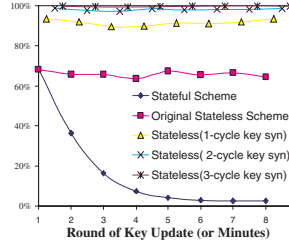
The communication overhead arises from several sources. First, each packet contains a PAH header, which introduces a message overhead. Second, the periodically broadcasted beacon message presents another source of communication overhead. Since the length of a beacon message is quite small (on the order of an IP header plus a PAH header), such a cost is tolerable. Finally, a node may have to retransmit the data packet, when it is key-unsynchronized with the receiver. In the worst case, a data packet may need to be retransmitted  $l$  times before it is successfully delivered to the destination, where  $l$  is the length of the path. However, since key un-synchronization generally does not happen frequently, the communication overhead caused by the packet retransmission is tolerable. In the next section, we present experimental results that investigate how frequently key-synchronization is required, and the percentage of retransmitted packets (extra communication delay).

**Performance.** We evaluated the performance of the proposed method by means of simulation. We created our own simulator, which implemented the transmission of data between nodes. For evaluation purposes, we assumed shortest path routing information was maintained at all times. This level of detail was sufficient to evaluate the effectiveness and performance tradeoffs of the proposed method.

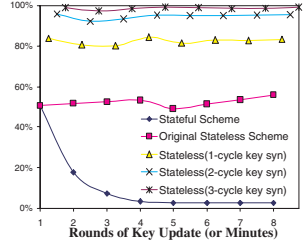
We generated a MANET with  $n$  nodes in a 1km x 1km area. Nodes were initially placed in a random way. Each node’s movement was simulated as a random walk with a



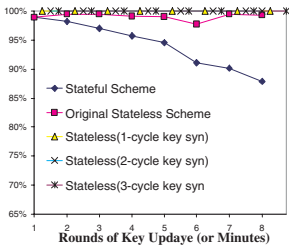
**Fig. 2.** Avg-percentage of reachable nodes from a random source node



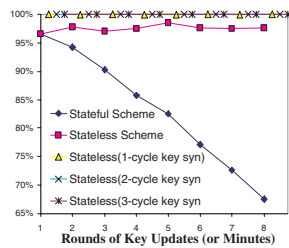
**Fig. 3.** Avg-percentage of nodes which received the latest key ( $P_{lost}=0$ , 40 Nodes)



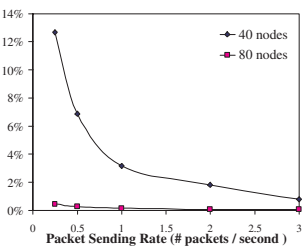
**Fig. 4.** Avg-percentage of nodes which received the latest key ( $P_{lost}=0.25$ , 40 Nodes)



**Fig. 5.** Avg-percentage of nodes which received the latest key ( $P_{lost}=0$ , 80 Nodes)



**Fig. 6.** Avg-percentage of nodes which received the latest key ( $P_{lost}=0.25$ , 80 Nodes)



**Fig. 7.** Percentage of data packets retransmitted due to key un-synchronization

maximum speed of 20m/s. The communication range for each node was uniformly set to 200m. Each data point measured is the average of 2000 simulations, using different seeds for random number generation. The average percentage of nodes with the latest session key was then measured after each round of key update for 8 consecutive rounds, supposing only the session key needs to be updated, and no revoked nodes.

The average percentage of reachable nodes from a randomly-selected source node, referred to as *reachability* in this paper, affects the performance of the proposed system. Figure 2 shows that for a fixed geographic area and transmission range, as the total number of nodes increases, the reachability increases. In the following experiments, we simulated a network with either 40 nodes, for which the reachability is 70%, or a network with 80 nodes, for which the reachability is 98%.

Next, we compared the proposed key synchronization scheme with the stateful and the original stateless key distribution schemes on the average percentage of nodes that have the latest session key after key updates. We assume the key manager updates the session key once each minute. The nodes in the proposed scheme broadcast a beacon message and synchronize session keys with neighbor nodes (if necessary) once every 15 seconds, i.e., there are 3 *cycles of key synchronization* between two consecutive key updates. Each cycle of key synchronization is assumed to complete before the next one begins. Data packets were not exchanged, so that only the effectiveness of key distribution was measured. The comparison results are shown in Figure 3 - 6.

From these figures, it is clear that a stateful rekeying scheme performs poorly in a MANET. After the key manager has updated the session key 6 times, almost no legitimate node can get the latest session key, even when they are still connected with the key manager. This is because a stateful scheme requires that a node receives every previous key update message, which becomes less and less likely as the number of updates increases. For the stateless rekeying scheme, the average percentage of nodes that have the latest session key is close to the average reachability. This is because successful key update is not dependent on receiving previous key updates. However, the limits on reachability still leave a substantial fraction of nodes that do not receive the latest session key. The proposed key synchronization scheme greatly improves the performance of key distribution, i.e., almost every node (99%) receives the latest session key after 3 cycles of key synchronization.

Figure 7 shows the packet retransmission ratio (due to key un-synchronization) for various packet sending rates, given a specific source and destination pair that communicate over a 100 minute period. It shows that (i) a denser network has a lower retransmission rate due to the lower probability of key un-synchronization, and (ii) the packet retransmission ratio decreases as the sending rate increases, since more data packets will be successfully transmitted after each key synchronization.

### 4 Implementation

We implemented and tested a prototype of the proposed system at the IP layer. Our implementation consists of two modules, the group rekeying module and the packet authentication module. The packet retransmission function is still under development.

The group rekeying module runs in the user space. It employs the stateless group key distribution scheme proposed in [8]. This module extracts the new session key and passes it to the packet authentication module via a kernel message. It also buffers the latest key update message for future key synchronization.

The packet authentication module runs in kernel space and is based on the Netfilter [1] architecture. It uses the HMAC-MD5 authentication algorithm,

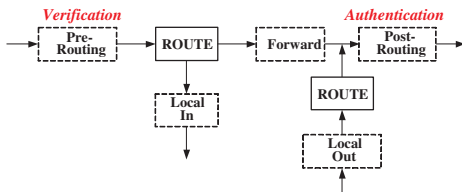


Fig. 8. Structure of Implementation on Netfilter

and will send a signal to the group rekeying module once receiving an unverified packet. Figure 8 illustrates how this module is implemented using Netfilter.

We tested our implementation on a small scale test bed consisting of a DELL Pentium IV laptop running Red Hat Linux 9.0 (kernel version 2.4.20), and two COMPAQ iPAQ 3970 PDAs running Familiar v0.7.2. (kernel version 2.4.19-rmk-pxa1-hh30). All of these were equipped with Lucent Orinoco wireless cards. The laptop acted as the key manager and the PDAs acted as regular nodes. We tested the functionalities of packet authentication, key distribution and synchronization, and revocation mechanism. All tests performed as expected and demonstrated that access control worked properly.

## 5 Related Work

Basagni, et. al [2] suggested using a symmetric key to secure the communication in a MANET. Their underlying idea is similar to ours. However, they suggested using hierarchical key management, and did not give a solution on how to synchronize the session keys and how to handle the unverified packets in the case of key un-synchronization.

Recently, Zhu, et. al., [16] designed a lightweight hop-by-hop authentication protocol (LHAP) for MANETs. LHAP uses the TESLA broadcast authentication [12] technique and the one-way key chain. It requires clock synchronization between nodes, and cannot completely prevent the propagation of forged packets. For instance, an attacker can use TESLA keys eavesdropped from a legitimate node *A* to authenticate its forged packets. The attacker can then fool nodes that set up a trust relationship with *A*, but lose contact for awhile, to accept these forged packets.

D. Kraft and G. Schafer [7] proposed a distributed access control scheme for consumer operated mobile ad hoc networks. Their scheme relies on a web-of-trust approach (like PGP [17]) for authentication. Such schemes are not designed for secure group communication, however, and therefore are not suited for our purpose.

Ioannidis, et. al., [6] presented a distributed firewall system. In their proposed system, a centrally defined security policy is propagated to each network endpoint, which execute this security policy to filter out unwanted packets. IPsec is used to authenticate users, protect traffic, and securely distribute credentials. This approach is likely to be impractical in a MANET, since IPSec is a point-to-point protocol, and managing a security association between every pair of nodes will be difficult.

R. Canetti, et. al., [3] proposed a host architecture for secure Internet multicast, which is somewhat similar to the proposed system. However, their architecture requires each member to set up an IPsec security association with the key manager, and does not solve the problem of key un-synchronization. Thus, it is impractical to be used as a network access control system for MANETs.

## 6 Conclusion and Future Work

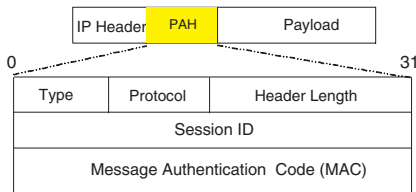
In this paper, we introduced a network access control system for MANETs based on stateless group key distribution. The system synchronizes keys when necessary for successful delivery, with proper access control. Simulation and a preliminary implementation demonstrate the proposed system is practical. Our future work includes designing a distributed key manager to eliminate this potential single point of failure.

## References

1. <http://www.netfilter.org>.
2. S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. Secure Pebblesets. In *Proc of MobiHOC 2001*, Long Beach, CA, 2001.
3. R. Canetti, P. Cheng, F. Giraud, and et.al. An IPSec-based Host Architecture for Secure Internet Multicast. In *Proceedings of NDSS'00*, San Diego, CA, USA, 2000.
4. D. Coppersmith and M. Jakobsson. Almost Optimal Hash Sequence Traversal. In *Proceedings of the Sixth International Conference on Financial Cryptography 2002*, 2002.

5. D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. *Advances in Cryptology-CRYPTO'02*, LNCS 2442:47–60, 2002.
6. S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a Distributed Firewall. In *Proceedings of CCS'00*, Athens, Greece, 2000.
7. D. Kraft and G. Schafer. Distributed Access Control for Consumer Operated Nobile Ad-hoc Networks. In *Proceedings of CCNC'04*, Las Vegas, Nevada, USA, January 2004.
8. D. Liu and P. Ning. Efficient Self-Healing Group Key Distribution with Revocation Capability. In *Proceedings of CCS '03*, Washington D.C., October 2003.
9. S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of Mobicom*, Boston, August 2000.
10. S. Mittra. Iolus: A Framwork for Scalable Secure Multicasting. In *Proceedings of ACM SIGCOMM' 97*, pages 277–288, Cannes,France, 1997.
11. D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. *Advances in Cryptology-CRYPTO'01*, LNCS 2139:41–62, 2001.
12. A. Perrig, R. Canetti, J. Tygar, and D. Song. Efficient Authentication and Signing of Multicast Stream over Lossy Channels. In *Proceedings of the 21st IEEE Symposium on Security and Privacy*, May 2000.
13. J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean. Self-Healing Key Distribution with Revocation. In *Proceedings of 2002 IEEE Symposium on Security and Privacy*, Berkeley, California, USA, May 2002.
14. D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. *IETF Request For Comments, RFC2627*, 1999.
15. C. Wong, M. Gouda, and S. Lam. Secure Group Communications Using Key Graphs. In *Proceedings of the ACM SIGCOMM '98*, pages 68–79, Vancouver, B.C, September 1998.
16. S. Zhu, S. Xu, S. Setia, and S.Jajodia. LHAP:A Lightweight Hop-by-Hop Authentication Protocol For Ad-hoc Networks. In *Proceedings ICDCSW'03*, Providence, Rhode Island, USA, May 2003.
17. P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Jun 1995.

## A Illustration of PAH



**Fig. 9.** Packet Authentication Header

Figure 9 shows the PAH in our implementation. The *Type* field specifies the type of packet, which is either a control packet or a data packet. Since the protocol type in the original IP header is replaced with an unsigned value (e.g., 199 in our implementation), its original value is held by the *Protocol* field in PAH so that it can be restored after the packet is accepted at the destination. The *Header Length* field indicates the length of the PAH. The *Session ID* field identifies the key session ID associated with

this packet. The *Message Authentication Code* (MAC) field holds the authentication information. Its length depends on the adopted authentication algorithm.

# Remotely Keyed Cryptographics Secure Remote Display Access Using (Mostly) Untrusted Hardware

Debra L. Cook, Ricardo Baratto, and Angelos D. Keromytis

Department of Computer Science, Columbia University, New York, NY, USA  
{dcook, ricardo, angelos}@cs.columbia.edu

**Abstract.** Software that covertly monitors user actions, also known as *spyware*, has become a first-level security threat due to its ubiquity and the difficulty of detecting and removing it. Such software may be inadvertently installed by a user that is casually browsing the web, or may be purposely installed by an attacker or even the owner of a system. This is particularly problematic in the case of utility computing, early manifestations of which are Internet cafes and thin-client computing. Traditional trusted computing approaches offer a partial solution to this by significantly increasing the size of the trusted computing base (TCB) to include the operating system and other software.

We examine the problem of protecting a user accessing specific services in such an environment. We focus on secure video broadcasts and remote desktop access when using any convenient, and often untrusted, terminal as two example applications. We posit that, at least for such applications, the TCB can be confined to a suitably modified graphics processing unit (GPU). Specifically, to prevent spyware on untrusted clients from accessing the user's data, we restrict the boundary of trust to the client's GPU by moving image decryption into GPUs. This allows us to leverage existing capabilities as opposed to designing a new component from scratch. We discuss the applicability of GPU-based decryption in the two scenarios. We identify limitations due to current GPU capabilities and propose straightforward modifications to GPUs that will allow the realization of our approach.

**Keywords:** GPUs, Encryption, Thin Clients, Video Conferencing.

## 1 Introduction

Spyware has been recognized as a major threat to user privacy. Especially when combined with a large-scale distribution mechanism (such as a popular web site or application, or a computer worm), the potential for large-scale security violations is considerable. Organizations increasingly spy on their employees' computer activities using the same technology, and public computers on Internet cafes are so riddled with such malware that only the most foolhardy of souls would use them for any sensitive application.

Work on addressing this problem has focused either on detection of spyware activity on a system or building a trusted system from the bottom-up, using a combination of hardware support, operating system extensions and application-specific logic. While



promising, these approaches offer only limited security against an adversary that legitimately controls the spyware-infected system, or against spyware that does not exhibit real-time activity (*e.g.*, consider a program that simply takes snapshots of the system's screen as the unsuspecting user is accessing some sensitive information). While images, like any data, can be sent encrypted over networks using existing protocols such as TLS and IPsec, decryption is performed by the operating system, creating the potential for the data to be copied by an untrusted client.

We propose to use the system's Graphics Processing Unit (GPU) as the only trusted component in our spyware-safe system for displays. By using GPUs, we leverage existing capabilities within a system as opposed to designing and adding a new component to protect information sent to remote displays. Sensitive content is directly passed to the GPU in encrypted form. The GPU decrypts and displays the content without ever storing the plaintext in the system's memory or exposing it to the operating system, the CPU, or any other peripherals. We use a remote-keying protocol to securely convey the decryption key(s) to the GPU, without exposing them to the underlying system. With this mechanism as our basic block, we can implement applications such as secure video broadcasts or remote desktop display access without trusting the rest of the system.

Our work is an initial step of which the main purpose is to propose the concept and determine the feasibility of GPU-based decryption. We determine that, with careful design, current GPUs allow for in-GPU image decryption at rates sufficient to support the example applications. We also identify several obstacles to fully implementing our scheme on current GPUs. The most difficult aspect of moving decryption into a GPU is the API and the types of operations supported within the GPU. [4] demonstrated that the APIs for GPUs are not designed to support operations typically found in symmetric key ciphers. As a result, we do not focus on forcing an existing symmetric key cipher to fit within a GPU in order to decrypt the data, but rather implement as many operations as possible within the GPU and confine the remaining ones to a *C* program in order to illustrate the concept. In the future, either a cipher suited for GPUs and/or support for additional operations in GPUs is required. We have begun work on a stream cipher designed for GPUs and include an estimate of the performance. We identify straightforward additions to future GPU designs that will allow for the realization of our scheme, and its possible integration with the Trusted Computing Group's proposed architecture.

## 2 Motivation

Applications to which our work is relevant include remote desktops (a thin-client scenario) and video conferencing displays. In a thin-client scenario, the client connects to a server which fulfills all of the client's computing needs [11]. Since all application logic is executed in the server, the client is completely stateless, and does little more than display updates sent by the server and forward local user input events. Current thin-client systems provide secure sessions by encrypting the display protocol before it is transferred over the network. However, in scenarios where the client terminal is untrusted, such as public computers, it may not be desirable for the host operating system to have access to the unencrypted display updates. Consider the system described in [8], wherein access to sensitive 3D data was controlled by manipulating the content sent to

the remote display client. Since the display data on the client cannot be secured, a number of additional mechanisms are devised to prevent the actual client application from being used as an attack tool on the system. In contrast, if the current display is only in decrypted form within the GPU, we only need to block reads by other applications.

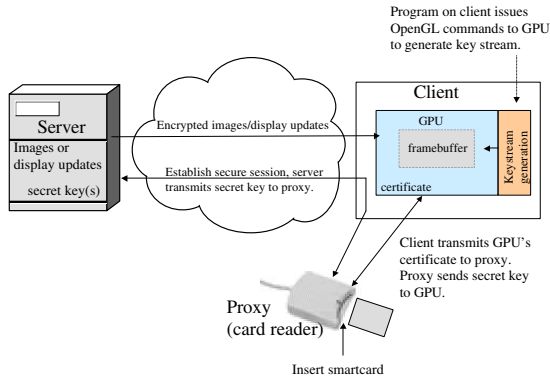
In video conferencing, we wish to prevent clients from copying the conference displays. How to secure video recorded at the client and audio is beyond the scope of this paper, although the concept we demonstrate with GPUs can also be applied to digital cameras and digital signal processors. While there are existing digital rights management (DRM) architectures aimed at preventing unauthorized copying of video, the images are still decrypted within the remote and untrusted OS. DRM includes how to manage the usage and trade of material [6], and must protect against both unauthorized access and unauthorized copying. An example is Microsoft's Windows Media Player DRM 9 Series, which includes the capability of authenticating and remotely-keying the media player [10]. The images are decrypted within the operating system by the media player then sent to the GPU. This architecture's security depends on using a specific closed-source media player and no program being able to access the memory utilized when decrypting the data. Alternative models of using trusted GPUs have been considered [2], but none has been implemented to our knowledge. The Trusted Computing Group's scope includes untrusted clients but its proposed architecture utilizes distinct trusted platform modules (TPMs), which may be hardware or software, to address multiple needs and provide a generic solution [14]. For graphical applications, our approach can be considered as an alternative that avoids specialized system components, or as a companion to TPMs. In particular, one possibility is for the TPM to handle key negotiation with the remote server, and then provide the session key to the GPU. We should note that similar concerns arise when handling voice traffic, as noted in [15].

Our main goal in moving decryption of graphics into the GPU is to prevent the underlying operating system or other software from gaining access to the unencrypted data. Specifically, we consider malicious software running on the client's operating system which attempts to read or modify displays and responses transmitted between the server and the client. We do not address modifications to the client's hardware, such as altering of the GPU. Furthermore, security of the client's surroundings (*e.g.*, a camera recording the client's display) is a separate problem outside the scope of our work.

## 3 Prototype

### 3.1 Architecture

Figure 1 depicts our overall architecture. A server encrypts the data and sends it to the client. The data remains encrypted until it enters the GPU where it is decrypted and displayed. The GPU's buffer is locked to prevent the display from being read by processes external to the GPU, effectively turning the frame buffer into a write-only memory. The decryption is performed via software running on the client's operating system which issues commands to the GPU (as opposed to a compiled program existing and executing entirely within the GPU's memory), with the operations performed within the GPU. This software does not have access to the keys and data contained inside the GPU; rather, it specifies the transformations (*i.e.*, decryption steps) that the GPU must



**Fig. 1.** Architecture for Remotely Keyed Decryption in the GPU

undertake. Ideally, any intermediate data produced by the decryption program, such as the keystream, are confined to the GPU. We explain in Section 4 why this is currently not possible with existing GPUs.

The decryption key changes on a per-session and application basis (and may even change within a session). Thus, the key must be conveyed to the GPU in a manner that prevents the client’s operating system from gaining access to it. One way to achieve this is to remotely key the GPU and decrypt the key therein. The key is used to generate the keystream directly within the GPU, exposing neither the key nor the keystream to the OS. The decryption of the key and generation of the keystream can be performed in a non-visible buffer (back buffer) on the GPU, to avoid visually displaying them. Reading the encrypted image into the back buffer with the logical operation of XOR enabled results in the image being decrypted. The result is then swapped to the front buffer to display the decrypted image to the user.

There are a few possibilities for how the entities involved are authenticated and how the key is sent to the GPU, depending on which components are trusted. In each case, it is assumed that the GPU contains a pre-installed certificate and private key. The certificate may be issued by the manufacturer and hardwired in the GPU. Another option is to allow writing the certificate to the GPU under circumstances when the client’s OS is trusted, such as when the GPU is first being installed on a newly configured client. The first and simplest option for authentication covers the case when the server sending the images is trusted and there is no need to verify the person viewing the images (*i.e.*, it is assumed that the fact the viewer was able to start the process on the client indicates it is safe to send the images) and/or the server is capable of authenticating a GPU based on its certificate. The server, either by establishing a session key with the GPU or using the GPU’s public key, encrypts the secret key and sends it to the GPU via the client. The second, more general scenario, also assumes the server is trusted but requires verification of the user viewing the images through a proxy entity, such as a smartcard reader. The user will activate the proxy by inserting a card into the smartcard reader attached to the untrusted system. The proxy will then establish sessions with both the server and remote system with the GPU. The server will convey the secret key to the GPU via the proxy, as shown in Figure 2. The process of converting the key from being encrypted under the server-proxy session key to being encrypted under

the proxy-GPU session key requires that the key be exposed only on the smartcard. The proxy and the GPU treat the underlying system, including the OS, as part of the network connecting them to each other and the server. A third scenario assumes that neither the server nor the client OS are trusted. When the images are encrypted, the encryption key is recorded on a smartcard. The encrypted images can then be stored on any server. To view the images on an untrusted system, the smartcard is inserted into a card reader (the proxy) or the key can be manually recorded and entered into the proxy. The proxy, using the GPU's public key, encrypts the secret key and sends it to the GPU via the client. The proxy does not have to be collocated with the client, but only has to be capable of exchanging information with the client. If a secret key only works for  $n$  blocks (such as  $n$  frames) of data, the remote keying will occur as needed to provide the key for each data segment.

The protocols used for the remote keying are not new. Refer to [1] and [5] for a discussion on authentication using smartcards. The novel component of our work is implementing one in a manner that avoids exposing the secret key outside the GPU. Any protocol used for the remote keying requires utilizing an asymmetric encryption algorithm to either encrypt the secret key directly with the GPU's public key or to establish a session key which is then used to encrypt the secret key. Obstacles arise due to the lack of support in GPUs for the operations required for public key ciphers, such as modular arithmetic for large integers. We discuss the limitations of the GPU in regards to public key cryptography when describing our prototype.

### 3.2 Implementation

To determine the feasibility of our scheme, we implemented the second scenario with 3 entities: a server, a proxy and the client. We use a stream cipher, RC4, to encrypt the images because of the rate of encryption required for streaming video. The prototype implemented as many operations as possible in the GPU via OpenGL, with the remaining operations restricted to a C program and which would be moved into a suitable GPU as we discuss in Section 4. Specifically, existing stream ciphers cannot be efficiently implemented entirely in OpenGL. We use the following notation:

- $K = k_1, k_2 \dots k_n$  is the set of secret keys used to encrypt the data.  $k_i$  encrypts the  $i^{th}$  subset of data. These keys may be individually pre-determined, or computed through a master key using a pseudorandom function.
- A frame refers to one frame of video or one display update.
- Rekeying refers to obtaining the next  $k_i$ . The interval at which rekeying occurs depends on either the number of frames displayed or the elapsed time.
- $r$  = is the number of frames or requests after which rekeying is required.
- $t$  = is the amount of time before rekeying is required.
- $sk$  = the session key used for communication between the server and proxy.
- $k^{pubk}$  = the GPU's public RSA key component.
- $k^{privk}$  = the GPU's private RSA key component.
- $m$  = the GPU's RSA modulus.

Figure 2 illustrates the steps for the remote keying and decryption of images in our prototype. A certificate containing a RSA [13] key is stored in the GPU's memory. For

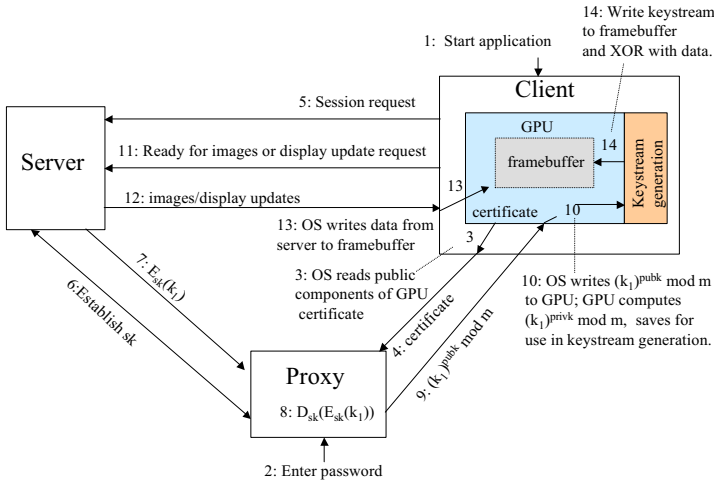


Fig. 2. Remotely Keyed Decryption in GPU Protocol Shown: logical links

our prototype, a program on the client uses OpenGL to write the certificate to the GPU then deletes it from the operating system’s memory. Installing a certificate in the GPU in this manner requires that the process be monitored to ensure that no program on the client gains access to the private key component of the RSA key while it is being written to the GPU. The certificate includes a public parameter containing an indication that the device is a GPU. When the application is started, the client’s OS reads the public information from the GPU’s certificate and sends it in a request to the proxy. The proxy, which requires activation either by entering a one-time password or inserting a smartcard, authenticates the GPU based on the information encoded in its certificate.

The client also sends a connection request to the server. The server contacts the proxy and a secure session is established between them. This can be accomplished using any protocol designed for secure session establishment. A single session key may be used for the entire session, or the session key can be changed periodically, depending on the protocol. In our prototype, the proxy authenticates the server based on the latter’s certificate, and uses a single session key,  $sk$ . When contacting the proxy, the server sends a random nonce and its certificate containing its public key for RSA. The proxy generates a random nonce, encrypts it with the server’s public key and sends it to the server. The server and proxy both concatenate the two nonces and use a hash of the result as  $sk$ . The server sends  $k_1$  encrypted with AES using key  $sk$  to the proxy. The proxy decrypts  $k_1$ , encrypts it with the GPU’s public key and forwards the result,  $k_1^{pubk} \bmod m$ , to the client. The client issues the OpenGL command to turn color mapping on then writes the value received from the proxy to a specific pixel location in the GPU. The color map corresponds to  $x^{privk} \bmod m$ , where  $x$  is the value being written, and results in decrypting the value from the proxy to obtain  $k_1$ . The write operation is done to the GPU’s back buffer to avoid visually exposing the resulting pixels (and annoy the user with unnecessary interference). As we explain later, we use a series of one-byte values for each  $k_i$ . The resulting pixels are used as the key to the stream cipher.

The client then signals to the server that it is ready to receive data or, for thin-client applications, makes a request to update a display. The server sends the encrypted data to

the client. Ideally, the GPU computes the keystream, writing the resulting bytes directly to the GPU's back buffer. As explained in Section 4, when using RC4 some  $C$  code is used to represent operations that will be performed in the GPU if improvements are made to the GPU's API. The client issues the OpenGL command to turn the logical operation of XOR on in the GPU, then writes the data received to the back buffer. The result is the data XORed with the keystream. The buffers are then swapped so the unencrypted image appears on the display. It is common practice to create an image in the back buffer then swap it to the front buffer in order to create a smooth transition between frames. After  $n$  frames or  $t$  time, the client must signal to the server that it needs the next secret key,  $sk_{i+1}$ , which is conveyed via the proxy as before.

Our prototype uses images encoded with 24 bits per pixel using 8 bits for each of the Red, Green and Blue components. No Alpha component is encoded since the image is written to the back buffer (which may not support the Alpha component) to be decrypted. The pixel format is a parameter used by certain OpenGL commands, such as the Draw command for writing data to the GPU, and can easily be changed to accommodate other pixel formats.

## 4 Design Decisions

We now discuss some of our design and implementation decisions that were guided by the constraints of existing GPUs. We first describe the limitations on programming a GPU to perform general keying and decryption operations, and then discuss the current inability to provide data compression.

GPUs are not designed to perform general arithmetic and byte-level operations. We refer the reader to [3] and [4] for background on GPU APIs and pixel processing, including the types of operations supported which are relevant to ciphers and the limitations of GPUs in performing byte level operations. There are no API commands for common operations such as modular arithmetic, shifts and rotates. Some operations can be performed by a sequence of other commands under certain circumstances, such as limiting values to a single byte and reading intermediate results from the GPU to the operating system to allow the result to be a parameter in a subsequent command. We describe how these limitations impact the ability to remotely key the GPU and decrypt data within the GPU, and the workarounds we used to create our prototype. We conclude that three enhancements to OpenGL are necessary to fully realize our architecture. First, a means of performing modular multiplication on values of magnitude typical of those used for public key ciphers is required to securely implement the remote keying. Second, a mechanism for using the contents of a pixel (or pixel component) as a parameter to an OpenGL command without first reading the pixel value from the GPU is required for the remote keying and keystream generation. Third, the ability to perform modular arithmetic using values less than 256 directly (*i.e.* without using color maps) is desirable to efficiently implement certain ciphers, such as RC4, within the GPU.

### 4.1 Remote Keying

The lack of modular arithmetic and limitations on the range of values in GPUs impacts the implementation of the asymmetric cipher used in the remote keying. The proxy

conveys the secret keys to the GPU via the client's OS using an asymmetric key cipher. Since existing public-key algorithms require exponentiation and/or modular arithmetic, the operations required cannot be emulated in the GPU with existing APIs, except when trivially small values are used, or when the values involved can be viewed as a series of 8 bits values. The remote keying of the GPU requires only that the GPU be able to perform the decryption function of the asymmetric algorithm. We note that unless the proxy and GPU share a secret key in advance, any protocol used to exchange information, whether by merely having the proxy encrypt information with the GPU's public key or by establishing a session key between them, requires use of an asymmetric cipher.

We considered two options for our prototype. First, the operations can be implemented in *C* code to represent a function that should be in the GPU. Second, restrictions can be imposed on the size of the asymmetric cipher's components to allow it to be implemented to run in the GPU. However, in the case of RSA this requires that plaintext and ciphertext each be restricted to fit in within a single byte, thus requiring the modulus and exponents also each fit within a single byte and resulting in key components too small to be secure. To illustrate the concept of decryption using public key cryptography within the GPU, we used "toy" values less than 256 in the prototype for the private and public exponents and the modulus. We used a series of 8-bit values to represent the data, *i.e.*, the secret key for RC4, encrypted with RSA. Each is encrypted with RSA by the proxy and sent to the GPU. When using RC4 as the keystream generator, up to 256 single-byte values can be in the series for RC4's secret key.

A third possibility is the integration of a decrypting GPU with a TPM such as the one proposed by the Trusted Computing Group. This chip could handle certificate storage and handling, as well a remote attestation and key negotiation. Our GPU can then handle image decryption using the TPM-negotiated session key.

## 4.2 Decryption of Data in the GPU

To decrypt the images received from the server, the GPU on the client must run a symmetric key cipher. As we described previously, we use a stream cipher. We consider two options for the stream cipher: using an existing stream cipher and designing a stream cipher suitable for a GPU. With respect to running an existing cipher within a GPU, operations typically found in symmetric key ciphers make this infeasible either due to the nature and number of OpenGL commands required to emulate the operations or due to the infeasibility to convert the operations to execute within the GPU given limitations of the API [4]. Existing stream ciphers, such as LILI, RC4, SEAL, SOBER and SNOW, are unsuitable for implementation in a GPU. We chose to use RC4 because it is possible to implement using OpenGL, though not practical due to the specific OpenGL commands required resulting in poor performance. The use of irregularly clocked feedback shift registers in LILI and SOBER, and 32-bit words in SNOW and SEAL, among other operations such as 9-bit rotations in SEAL, make these either less attractive than implementing RC4 or impossible to implement in OpenGL.

The operations in RC4 consist entirely of adding two bytes, modulo 256 and swapping two bytes. Thus, the only operation required of RC4 which is lacking in a GPU is modular arithmetic. Since the modulus is 256, all values can be represented by single bytes and can be stored as individual pixel components. Given two integers,  $a$ ,  $b$  in

the range  $[0,255]$ ,  $a + b \bmod 256$  can be computed using a color map. This requires knowing either  $a$  or  $b$  in advance to determine which color map to activate. For each integer,  $a$ , in the range  $[0,255]$ , create a color map where the  $i^{\text{th}}$  entry corresponds to  $a + i \bmod 256$ . To compute  $a + b \bmod 256$ ,  $b$  is stored as a pixel component, the color map for  $a$  is activated, then the pixel containing  $b$  is copied to a new location. The result written to the new location will be the  $b^{\text{th}}$  entry of the color map. This poses two problems. First, while OpenGL is used, the command to activate a color map must be issued by a program running on the operating system, requiring  $a$  to be exposed to the operating system. While this does not expose the keystream to the OS, it does provide partial information to the operating system, which may be helpful in determining keystream values. Second, the copying of pixels between locations in the buffer is one of the slowest operations within GPUs. In addition to the copy needed to compute the sum, copies are needed to update the indices and move bytes into the appropriate pixel components and locations. As a result, implementing RC4 in OpenGL is not a practical option. Therefore, we opted to implement the keystream generator of RC4 in  $C$  to represent a function that will eventually be moved into the GPU. The keystream bytes are written to the GPU as they are computed. This requires the  $C$  function computing the keystream to read the secret key from the GPU. We initially wrote each byte of output from RC4 directly to the GPU as it was generated. However, the number of writes required (750,000 for a 500x500 image) resulted in poor performance. We changed our prototype to compute the keystream bytes for an entire row of pixels before writing them to the GPU, reducing the number of writes to the height of the image with the tradeoff that a segment of the keystream is temporarily stored outside the GPU.

Due to the inability to efficiently generate a keystream within a GPU by using an existing stream cipher, we are investigating designing a stream cipher utilizing graphics operations for which GPUs are designed. We briefly describe the concept here. By mapping a texture exhibiting sufficient randomness to a continuously morphing image while changing certain variables, such as viewpoint and lighting, and extracting pixels from the image, a keystream is generated. The keystream is never within the client's system memory in this case. We experiment with an initial version in order to estimate the time to compute the keystream, with the results shown in Section 5. We point out that while creation of a new stream cipher suitable for current GPUs is feasible (and in fact may have wider applicability than our applications), the same is not true for public-key ciphers, since this would require devising a new one-way function that does not require exponentiation and modular arithmetic on numbers larger than a single byte.

While the proposed approach protects the secrecy of the images sent to the untrusted system, the integrity of these images is not protected. This could allow an attacker to change parts of the image, although this would be immediately detectable by the user, as it would produce corrupt output on the screen (since the attacker does not know the session key). Adding a message authentication code (MAC) to our scheme is not currently feasible due to the limits of current GPUs.

## 5 Experiments

We conducted two sets of experiments to measure the ability of current GPUs to sustain decryption rates compatible with our example applications. We used OpenGL as



the API to the graphics card driver. We did not use any vendor-specific OpenGL extensions, making our prototype GPU-independent. We used GLUT to open the display window. The only requirement is that the GPU must support 32-bit “true color” mode, as the routine for decrypting the secret key requires representing bytes in a single-pixel component. The code for the client consists of C, OpenGL and GLUT, compiled using Visual C++ version 6.0. The processes for the server and proxy are written in JAVA.

The experiments utilized three different clients in order to test different GPUs. The environments were selected to represent a fairly current computing environment, a laptop and a low-end GPU. In all cases, the display was set to use 32-bit true color with full hardware acceleration. The clients are:

1. A Pentium IV 1.8 GHz PC with 256KB RAM and an Nvidia GeForce3 Ti200 graphics card with 64MB of memory, running MS Windows XP. The GPU driver uses OpenGL version 1.4.0.
2. A Pentium Centrino 1.3 GHz laptop with 256KB RAM and an ATI Mobility Radeon 7500 graphics card with 32MB of memory, running MS Windows XP. The GPU driver uses OpenGL version 1.3.425.
3. A Pentium III 800 Mhz PC with 256KB RAM and an Nvidia TNT32 M64 graphics card with 32MB of memory, running MS Windows 98. The GPU driver uses OpenGL version 1.4.0.

We simulated streaming video applications, such as NetMeeting, by sending a stream of images from the server to the client. We tested with frame sizes of 320x240 and 500x500 pixels. The frames were encrypted and stored in individual files on the server prior to starting the application. To measure thin-client performance, we used the average update size of 2,112 pixels (a 16x132 pixel area) from the standard i-Bench [7] web benchmark for thin-clients. The update sizes in i-Bench range from 1x1 areas to 1,007x622 areas (626,354 pixels). All tests used images encoded as 24-bit RGB pixels, with 8-bits per color component.

For each image size, two types of tests were run. The first set of tests determined the delay due to the additional computation needed for the remote keying and decryption, compared to sending unencrypted images. In these tests, all three entities (server, proxy, and GPU) were run on the same PC or laptop. Each of the three clients was tested. The results of the first set of tests are shown in Figure 3.

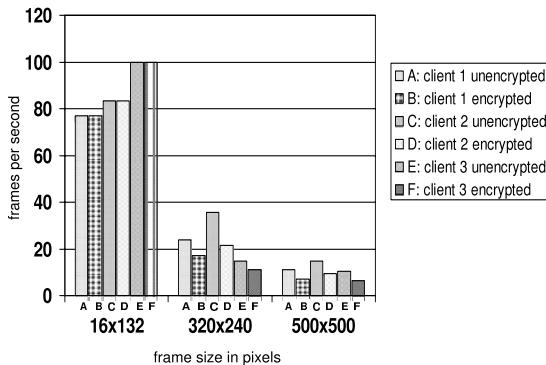


Fig. 3. All Entities on a Single System

The second set of tests involved running each entity on separate systems on a LAN to determine the overall performance when the data arrival rate was impacted by network delay. The first client with the Nvidia GeForce3 GPU was used for these tests. Figures 4 and 5 show the results of these experiments. Two tests were run using two different LANs. In one case, the server and proxy were dedicated to the experiment and there was no traffic leaving the server and proxy aside from that due to our experiment. In the second case, we ran our tests on shared servers used for general purpose computing. In both cases, each element had a 100Mbps connection to the LAN. There were three hops between the client and server, and between the client and proxy; there are two hops from the proxy to the server.

For all tests, the number of frames per second (fps) for both encrypted and unencrypted frames are provided. In video conferencing applications, the fps supported is important: a minimum rate of 10 fps is required to obtain tolerable video and is typical in such applications, with 24 fps and higher rates required for better quality. In contrast, the rate of updates in thin-client applications is dependent on user requests and will be sporadic. The fps reflects the maximum supported burst rate.

We note that it was not our intention to build a robust streaming video application using RTP which accounted for delay, rate of transmission and lost packets, but rather we focus on the remote keying and decryption within the GPU, and determine the resulting overhead. Therefore, TCP was used for all communication between the entities.

At least 99% of the delay when decrypting frames with RC4, compared to using unencrypted images, is due to the writing of the keystream bytes to the GPU. The keystream was written to the GPU one row at a time. When the test is run with the write eliminated (all other operations for the decryption are still performed), the average time is the same as that for the unencrypted images. The actual computation of the keystream per frame, enabling the logical operation of XOR in the GPU and swapping of buffers takes less than *1ms* for the 500x500 frames on all clients. When testing the average thin-client display size update (2,112 pixels), the times for the encrypted updates were the same as for the unencrypted updates because the keystream required only 16 writes to the GPU. In contrast, the 320x240 and a500x500 pixel frames required 240 and 500 writes per frame, respectively.

The limiting factor in the processing of the 2,112-pixel updates is the time for the server to create the update (read the update from a file in our experiment). To determine the rate at which the client can process such updates if creation of updates is not a limiting factor, an array containing 2,112 pixels was stored in memory on the server

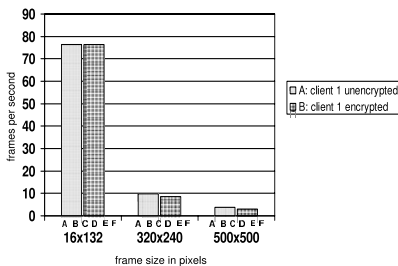


Fig. 4. Dedicated Lan and Client 1

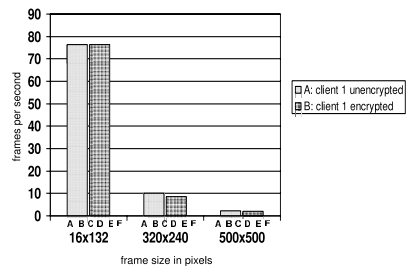


Fig. 5. Shared Lan and Client 2

and repeatedly sent to the client. The client can process over 500 updates per second on each of the three platforms, indicating that decryption overhead and the GPU are not limiting factors for small updates. For larger updates in thin-client applications, we do not consider an increased delay, *e.g.*, when the entire display changes, to be an issue; such updates are infrequent and, from a human perspective, are no worse than the loading of some web pages or opening of applications.

When sending images over a LAN, the decreased rate for the 320x240 and 500x500 pixel frames compared to the case when all processes were on the same PC is due to the rate at which images are sent from the server to the client being limited by the bandwidth. Even if no bandwidth is consumed by protocols, a maximum of 16.66 uncompressed 500x500 RGB frames can be transmitted per second on a 100Mbps interface.

To estimate the time required for computing a keystream designed for the GPU as described at the end of Section 4, we loaded an initial image in the GPU and measured the time to execute all of the OpenGL operations under consideration. After each series of executions, the resulting image is the keystream XORed with the current encrypted frame. The execution per frame is less than 1ms, indicating that any differences in the time to process encrypted *vs.* unencrypted frames will be imperceptible.

The time for the remote keying is mainly dependent on the time to enter the password or insert the smartcard into the proxy, and may take up to a few seconds if a password must be entered. Aside from this, the time is dependent on the protocol used and on the transport delay between the entities. Using a public-key encryption algorithm, generating random nonces and encrypting the secret key with AES requires approximately two seconds in each environment.

## 6 Conclusions

We addressed the feasibility of decrypting images and displays within a graphics processing unit as a way of combating the rising threat of spyware. Our primary insight is that a suitably modified GPU can serve as a minimal trusted computing base for displays in certain types of widely used applications, such as video conferencing and remote desktop display access. The main mechanism in our scheme is decryption of frames exclusively inside the GPU, without storing either the key material or the plaintext on the system's main memory. Our technique can protect against many types of spyware, as well as several attacks aimed at the human interface layer [9].

We explained why this scheme cannot fully be realized due to current limitations of GPU APIs. We identified three straightforward enhancements to GPU APIs that can overcome these limitations. With our prototype, we demonstrated that the concept is feasible for thin-client applications and the video broadcast in conferencing applications. Designing a keystream which runs entirely in the GPU and takes advantage of typical graphics operations will eliminate overhead and improve performance. To further improve performance in these applications, image compression facilities will need to be implemented inside the GPU, a trend which is already occurring. In addition, our numbers show that for typical video conferencing frame rates and web browsing using thin-clients, the lack of compression is not a performance bottleneck.

Our prototype focused on the securing of images sent to an untrusted client. Some additional items must be considered in a complete system that protects all inputs. For

example, protecting any user keyboard and mouse inputs on the client which must be conveyed to the server. Also, depending on the application, audio may need to be encrypted in a manner that prevents the OS from accessing the plaintext. The types of operations supported by programmable DSPs make extending our concept to audio relatively easy. We refer the reader to the extended version of this paper [3] for a complete discussion. Other items discussed in [3] include proxy attacks in relation to our model, data compression when using GPU based decryption and server-side encryption when using a GPU based stream cipher. Future work includes developing prototypes that fully integrate the concept into thin-client applications and expanding the prototype to include encryption within DSPs.

## References

1. M. Abadi, M. Burrows, C. Kaufman, B. Lampson, Authentication and Delegation with Smart-cards, *Theoretical Aspects of Computer Software*, 1991.
2. P. Biddle, M. Peinado and D. Flanagan, Privacy, Security and Content Protection, <http://download.microsoft.com/download/a/ff/c/afcf8195-0eda-4190-a46d-aa60b45e0740/Secure.ppt>
3. D. Cook, R. Baratto and A. Keromytis, Remotely Keyed Cryptographics - Secure Remote Display Access Using (Mostly) Untrusted Hardware (Extended Version), Columbia University Computer Science Technical Report CUCS 050-04, 2004.
4. D. Cook, J. Ioannidis, A. Keromytis and J. Luck, CryptoGraphics: Secret Key Cryptography Using Graphics Cards, *RSA Conference, Cryptographer's Track (CT-RSA)*, LNCS 3376, Springer-Verlag, pages 334-350, 2005.
5. H. Gobiuff, S. Smith, J. Tygar and B. Yee, Smart Cards in Hostile Environments, *2nd USENIX Workshop on Electronic Commerce*, 1996.
6. R. Iannella, Digital Rights Management (DRM) Architectures, *D-Lib Magazine*, <http://www.dlib.org/dlib/june01/iannella/06iannella.html> vol. 7 (6), June, 2001.
7. i-Bench version 1.5, Ziff-Davis, Inc, <http://www.veritest.com/benchmarks/i-bench/>.
8. D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Crocchia and P. Cignoni and R. Scopigno, Protected Interactive 3D Graphics Via Remote Rendering, *ACM SIGGRAPH*, 2004.
9. E. Levy, Interface Illusions, *IEEE Security & Privacy*, vol. 2 (6), 2004, pages 66-69.
10. Microsoft Windows 9 Media Series Digital Rights Management, <http://www.microsoft.com/windows/windowsmedia/drm.aspx>, 2004.
11. J. Nieh, S. Jae Yang and N. Novik, Measuring Thin-Client Performance Using Slow-Motion Benchmarking, *ACM Transactions on Computer Systems*, vol. 21 (1), pages 87-115, 2003.
12. OpenGL Organization, <http://www.opengl.org>.
13. RSA Laboratories, PKCS #1: RSA Encryption Standard Version 1.5, November, 1993.
14. Trusted Computing Group, Trusted Computing Group Architecture Overview, <https://www.trustedcomputinggroup.org/home>, 2004.
15. T. J. Walsh and D. R. Kuhn, Challenges in Securing Voice over IP, *IEEE Security & Privacy Magazine*, vol. 3 (3), May/June 2005, pages 44-49.
16. M. Woo, J. Neider, T. Davis and D. Shreiner, *The OpenGL Programming Guide*, 3rd edition, Addison-Wesley, Reading, MA, 1999.

# Authenticating Query Results in Data Publishing

Di Ma<sup>1,3</sup>, Robert H. Deng<sup>2</sup>, Hweehwa Pang<sup>2</sup>, and Jianying Zhou<sup>3</sup>

<sup>1</sup>University of California, Irvine

{dma1}@uci.edu

<sup>2</sup>Singapore Management University, Singapore

{robertdeng, hhpang}@smu.edu.sg

<sup>3</sup>Institute for Infocomm Research, Singapore

{madi, jyzhou}@i2r.a-star.edu.sg

**Abstract.** We propose a communication-efficient authentication scheme to authenticate query results disseminated by untrusted data publishing servers. In our scheme, signatures of multiple tuples in the result set are aggregated into one and thus the communication overhead incurred by the signature keeps constant. Next attr-MHTs (tuple based Merkle Hash Tree) are built to further reduce the communication overhead incurred by auxiliary authentication information (AAI). Besides the property of communication-efficiency, our scheme also supports dynamic SET operations (UNION, INTERSECTION) and dynamic JOIN with immunity to reordering attack.

**Keywords:** data publishing, authentication, merkle hash tree, aggregated signature.

## 1 Introduction

### 1.1 The Third Party Publishing Problem

This paper studies techniques for authenticating query results in the third party publishing scenario shown in Figure 1, where database owners outsource their data management to a third party publisher to disseminate data to users on demand. The motivation of outsourcing is to achieve greater data survivability and higher distribution efficiency: Firstly, as external servers take care of the data management, organizations can concentrate on their core tasks and thus reduce substantial cost on software, hardware and hiring professionals to maintain the in-house system; Secondly, by adding data processing server(s) near user cluster, users can get faster response to their queries from the server; Last, secret keys which are used for protection of the database are kept on the corporate end and not online, so that much better security is achieved.

As valuable information is stored in the third party publishing servers, the servers as well as the data delivery networks are frequently the targets of malicious attacks. Furthermore, the server itself might be malicious. A malicious server may attempt to insert fake records into the database or modify existing records. As a result, the integrity and origin authenticity of query results coming from these servers must be verified before a querier can consume them. Especially when the querier is dependent on the results to make high-stake decisions, she needs strong guarantees of the integrity and accuracy of the data received.

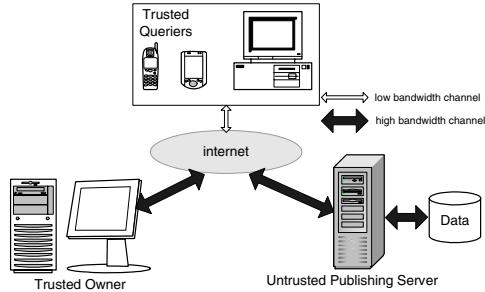


Fig. 1. System Set-Up

## 1.2 Related Work and Our Contributions

We propose a novel scheme to authenticate query results disseminated by an untrusted third party publishing server. Our scheme is based on the Merkle Hash Tree (MHT) and aggregated digital signatures. A couple of database authentication schemes employing MHT (hereafter referred to as MHT-based schemes) are proposed in the literature [7,12]. However, the way we make use of MHT is unique. The MHT in previously proposed schemes is constructed over the entire relation, while our MHT (referred to as attr-MHT) is constructed on individual tuples in order to reduce communication overhead incurred by auxiliary authentication information (AAI). In our view, the disadvantages of constructing a MHT over the entire relation are as follows: 1) it is very expensive to perform data update as any change in the relation will have impact on the whole MHT; 2) as a result of 1), it is not suitable to authenticate frequently changing data such as stock and sales information; 3) it is difficult to reduce the communication overhead incurred by AAI and imposes high processing cost on the querier (e. g., PROJECT has to be performed by the querier [7]); and 4) it does not support dynamic JOIN and SET operations.

Our research is motivated by the work done by Mykletun, et. al. [10] (hereafter we refer to the scheme in [10] as well as our scheme as the aggregated signature based schemes) which explores the use of aggregated signature schemes in database system authentication. Their main contribution is the reduction of communication overhead incurred by signatures, through an aggregated signature scheme. In this paper, we extend and improve their work by constructing attr-MHTs over individual tuples to reduce the communication overhead of AAIs that result mostly from PROJECT operations and applying to all the relational operations. By using MHT and aggregated signature scheme synergetically, our scheme is the first authentication scheme which reduces communication overhead incurred by AAI as well as supports dynamic JOIN and SET operations. The main contributions of our scheme are: 1) achieve constant communication overhead for SELECT and allow a querier to verify all the result tuples with just one digital signature verification operation; 2) minimize communication overhead incurred by AAI for PROJECT operation through optimized attr-MHT; 3) support dynamic SET operations (UNION, INTERSECTION); 4) support dynamic JOIN with immunity to reordering attack.

## 2 Preliminaries

### 2.1 The Merkle Hash Tree

We illustrate the construction and application of the MHT with a simple example. The reader is referred to [9] for detailed description. To authenticate data values  $n_1, n_2, \dots, n_w$ , the data source constructs the MHT as depicted in Fig. 2 assuming that  $w = 4$ . The values of the four leaf nodes are the message digests,  $H(n_i)$ ,  $i = 1, 2, 3, 4$ , respectively, of the data values under a one-way hash function  $H(\cdot)$ . The value of each internal node of the tree is derived from its child nodes. For example, the value of node  $A$  is  $h_a = H(H(n_1) \| H(n_2))$  where “ $\|$ ” denote concatenation. The value of the root node is  $h_r = H(h_a \| h_b)$  which is used to authenticate any subset of the data values  $n_1, n_2, n_3, n_4$ , in conjunction with a small amount of AAI. For example, a user, who is assumed to have the authentic root value  $h_r$ , requests for  $n_3$  and requires the authentication of the received  $n_3$ . Besides  $n_3$ , the source sends the auxiliary information  $h_a$  and  $H(n_4)$  to the user. The user can then check the authenticity of the received  $n_3$  as follows. The user first computes  $H(n_3)$ ,  $h_b = H(H(n_3) \| H(n_4))$  and  $h_r = H(h_a \| h_b)$ , and then checks if the latter is the same as the authentic root value  $h_r$ . Only if when this check is positive, the user accepts  $n_3$ . The concept of MHT has been used for certifying query answers over XML documents [6], proving the presence/absence of public key certificates on revocation lists [8,11], certifying data published by untrusted publishers [7] and certifying JPEG2000 sub-images [5].

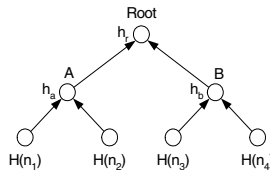


Fig. 2. An example Merkle Hash Tree

### 2.2 Aggregated Signature Scheme

A digital signature algorithm is a cryptographic tool for generating non-repudiation evidence, for authenticating the integrity of the signed message as well as its origin. An aggregated signature scheme is a digital signature scheme which allows aggregation of multiple individual signatures into one *unified* signature such that verification of the unified signature is *equivalent* to verifying individual component signatures. The concept of aggregated signature is first introduced by Boneh, et al. in [2] (hereafter referred to as BGLS scheme). According to the ability to aggregate signatures from different signers, there are single-signer scheme and multi-signer scheme. Single-signer scheme, like Condensed-RSA [10], aggregates only signatures from the same signer into one unified signature. Multi-signer scheme, like BGLS, can aggregate signatures from multiple signers.

### 3 System Overview

Our proposed authentication scheme is targeted for the third party publishing model [7] depicted in Fig. 1. The objective of our scheme is to provide adequate security measures to protect the stored data from both the malicious outsider attacks and the server itself.

#### 3.1 Trust Model

The owner of data is fully trusted by the queriers. Only the owner knows the private key for signing individual tuples. Queriers are trusted database clients and a querier verifies query results by checking the data integrity and data origin based on the owner signatures. The data processing server is responsible for replication, backup and dissemination of the outsourced database that it hosts. However, the server is not trusted with the integrity of the data.

There are two kinds of attacks: server side attacks and communication channel attack. Server side attacks refers to attacks happening on the server side. The server is assumed to be unsecured, meaning it is possible for a hacker to tamper (insert, delete, modify) with the data there. Also, the server itself might be malicious and it may attempt to tamper with the data it stores or processes. Insertion attack and modification attack of the data on the sever side can be detected easily at the querier side. Deletion attack is a bit more complex. There are two kinds of deletion attacks happening on the server side: permanent deletion of tuples from the database and dynamic deletion of tuples from the result set (the database remains intact). Dynamic deletion cannot be detected by the querier. It is an unsolved problem in the literature [7,12,10] and stays unsolved in our scheme. However we have some methods to detect permanent deletion attack. One way to detect permanent deletion attack is for the DBMS to maintain a global information map (such as a MHT on the entire database), and to check the integrity of the database periodically. How to maintain the global information map for the entire database is beyond the scope of this paper. Our strategy is putting the responsibility of integrity check of the whole database on the owner side. Although the capability provided by permanent deletion detection is limited but still useful, in the sense that users can be sure that they are working with authentic data, although they cannot be sure they can receive all the relevant data.

When the querier and the server are not communicating over a secure channel, e.g., SSL/TLS, the open communication channel is totally untrusted and the query result set is vulnerable to communication channel attacks such as deletion of records, insertion of spurious records, modification of valid records and reordering of query results from JOIN. Any communication channel attacks can be detected at the querier side.

#### 3.2 Overview of System Operation

Using our scheme, the system in Fig. 1 operates as follows:

1. An owner prepares an authenticated tuple in the database by constructing a MHT on the tuple. As the leaves of our MHT are the digests of attribute values in the



tuple, we refer to such a MHT as attr-MHT to distinguish it from the MHT constructed on the entire table. The owner signs on the root value of the attr-MHT. The authenticated table, consisting of all the tuples and their corresponding signatures, are then uploaded to the publishing server(s) through a high bandwidth channel.

2. The server executes a client's query by selecting records that match the query predicate, generates a verification object  $VO = \{\sigma_{1,t}, \Lambda\}$  and sends back the  $VO$  along with the result set. A verification object is an object which contains enough information for the client to verify the query result set. The  $VO$  consists of two parts. The first part  $\sigma_{1,t}$  is a unified aggregated signature calculated from all the tuple signatures in the result set. The second part  $\Lambda$  represents authentication auxiliary information (AAI) which is necessary to authenticate the result set.
3. The client, or the querier, after receiving the result set and the  $VO$ , verifies the authenticity of the result set using the  $VO$ . As the queriers may have a diversified range of computation and communication capabilities, an important design objective of our authentication scheme is to minimize communication and computation overhead at the querier side.

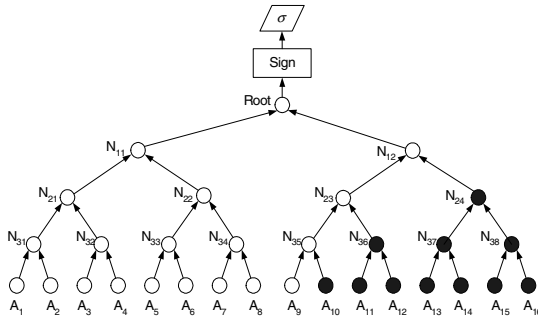
## 4 The Scheme

First, we show how to construct attr-MHT and generate tuple signature over it. Next we illustrate how to aggregate tuple signatures over a query result set and discuss considerations in choosing one of the aggregated signature schemes. We then present details on how to construct  $VO$ s of result sets from various basic algebraic operations [3].

### 4.1 The attr-MHT and Calculating Tuple Signature

An attr-MHT is a MHT built on an individual tuple. We use an example to illustrate the construction of an attr-MHT. Suppose there is a table with 16 attributes, for each tuple in the table, we construct an attr-MHT. Fig. 3 shows a binary attr-MHT constructed from one tuple in the table. Though the attr-MHT in our example is a binary and balanced tree, in general it can be non-binary and unbalanced. In the tree of Fig. 3, a leaf node corresponds to an attribute in the tuple and is assigned a value which is the message digest of the attribute value of this tuple. The internal nodes are assigned with values derived from its two child nodes. This process continues until the root value  $h$  is computed. The owner generates a tuple digital signature  $\sigma$  on the root value  $h$ :  $\sigma = SIGN(h)$ . To verify the authenticity of this tuple, the querier reconstructs the attr-MHT to compute the root value and then verifies the digital signature using the owners' public key and the computed root value. The querier accepts the received tuple as authentic only if the verification is successful.

The purpose of using attr-MHT in our authentication scheme is to reduce the communication overhead incurred by AAI which is used to verify a record from a PROJECT operation. For example, without using attr-MHT, if attributes  $A_{10}$  to  $A_{16}$  are filtered out, to authenticate record  $\langle A_1, A_2, \dots, A_9 \rangle$ , the server needs to send seven hash values (the hash values of nodes  $A_{10}$  to  $A_{16}$ ) to the querier. With the help of attr-MHT, only three hash values (the hash values of nodes  $A_{10}$ ,  $N_{36}$  and  $N_{24}$ ) need to be sent out to



**Fig. 3.** An example attr-MHT

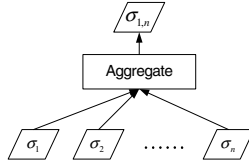
the querier. More details on how to optimize the attr-MHT to further reduce the size of AAI are discussed in Section 4.3.

When there exist multiple relations in the database, to identify a tuple in a specified relation, the owner generates a digital signature for this tuple with the relation’s name. The signature generated with relation’s name is represented as  $\tilde{\sigma} = SIGN(\tilde{h})$ , where  $\tilde{h} = H(h||RELATION\_NAME)$  ( $h$  is the root value of the tuple attr-MHT). For example, for a tuple in a relation named “R”, its signature with relation’s name is calculated as:  $\tilde{\sigma} = SIGN(H(h||“R”))$ . After generating all the tuple signatures in the database, the owner uploads the authentic database (the original database and all the tuple signatures) to the server.

**4.2 Aggregating Tuple Signatures**

A query result set usually includes tens, hundreds or even thousands of records. To verify these records, a straightforward solution is that the server sends one signature per record to the querier and the querier verifies these records one by one. This direct solution is not very attractive on two counts. First it is not efficient because it is expensive to verify these records one by one. We thus favor a solution that enables the querier to do batch-verification: verify once and verify all the records. Furthermore, as the size of a database record can be small (relative to the signature and digest) in many cases, the communication overhead incurred by tuple signatures is not negligible. Aggregated signature schemes provide a near perfect solution. Mykletun, et. al. [10] is the first to use aggregate signature schemes to authenticate query results from out-sourced databases under three system models, i.e., Unified-Client model, Multi-Querier model, and Multi-Owner model, to reduce communication overhead. Suppose there are  $n$  tuples in the result set, the server aggregates these tuple signatures. The aggregation process is depicted in Fig. 4.2 where  $\sigma_i$  denotes the signature of tuple  $i$  in the result set and  $\sigma_{1,n}$  denotes the aggregated signature of the result set. The aggregated signature  $\sigma_{1,n}$ , instead of the  $n$  individual tuple signatures, is sent to the querier as part of VO for query result verification.

There are some concerns in choosing one of the two aggregated signature schemes to be used in an authentication scheme. First, bandwidth cost concern. The Condensed-



**Fig. 4.** Aggregating tuple signatures

RSA scheme achieves constant bandwidth in both the Unified-Client and Multi-Querier models but not in the Multi-Owner model while the BGLS scheme achieves constant bandwidth in all the three models.

Second, verification cost and system model concern. The verification costs of both schemes are linear to the number of signatures. The Condensed-RSA scheme is not suitable for the Multi-Owner model, that is, tuples in a database are signed by different signers who create them. For example, a McDonalds database would have sales information from each franchise and each franchise could sign its own sales data. The BGLS signature scheme can aggregate signatures by distinct users into one short signature, however the computational complexity is unfortunately quite high Compared with the Condensed-RSA scheme.

In a Multi-Owner scenario, the BGLS scheme is more favored since it can aggregate any two or more signatures from different signers. Although Condensed-RSA cannot aggregate signatures from different signers, to achieve more computation efficiency, it still can be used in our authentication scheme with a slight modification. If we regard the whole database as a property of an organization, every tuple is owned by the organization and should be organization-signed. In a multi-signer application, if we choose to use organization-signed tuple signatures instead of individual creator signed tuple signatures, Condensed-RSA can be used in the same way as the BGLS is used. In the rest of the paper, we use BGLS as the default signature scheme to illustrate our authentication scheme.

**4.3 VOs for Relational Algebraic Operations**

In the relational data model, queries against a database are formulated in SQL [4]. Such queries are then typically translated by the DBMS query processing engine into expressions of the relational algebra for the purpose of query optimization and execution. We mainly concerns about providing VO for query results where the queries are formulated as expressions of relational algebra. We will illustrate in this Section how to construct VOs for various relational algebraic operations: SELECT, PROJECT, SET operations (UNION, INTERSECT), and JOIN. They are the basic algebraic operations in a relational database system. Although we present a solution for each operation separately, the solutions for individual operations can be combined together to provide a solution for complicated queries, e.g. a SELECT-PROJECT-JOIN query.

**VO for SELECT.** SELECT is defined as:  $\sigma_C(R) := \{t|t \in R \text{ and } C(t)\}$  where  $R$  is a relation,  $C$  is a condition of the form  $A_i \Theta c$  where  $A_i$  is an attribute of  $R$ ,  $c \in D_i$ ,  $D_i$

is the domain on which attribute  $A_i$  is defined, and  $\Theta \in \{=, \neq, <, >, \leq, \geq\}$ . It extracts specified tuples from a target relation.

With the definition of  $VO$  given in Section 3, the use of aggregated signature to construct a  $VO$  is straightforward. Suppose there are  $t$  tuples in the result set, the  $VO$  for this result set is:  $VO = \{\sigma_{1,t}, \phi\}$ , where  $\sigma_{1,t}$  is the aggregated signature calculated from signatures of the  $t$  tuples in the result set and  $\phi$  denotes null. To verify the result set, the querier reconstructs the attr-MHT for each tuple in the result set and calculate the root value of the attr-MHT:  $h_i, i = 1, 2, \dots, t$ . Next the querier verifies the query result with  $\sigma_{1,t}$  and the calculated  $h_i$ .

**$VO$  for PROJECT.** PROJECT is defined as:  $\pi_{A_k, \dots, A_l}(R) := \{\langle t.A_k, \dots, t.A_l \rangle | t \in R\}$ . It extracts specified attributes from a target relation. For queries involving PROJECT operations, some attributes of the tuple are filtered out. In this case, additional information need to be sent to the querier to reconstruct the attr-MHT. Suppose there are  $t$  records in the result set, the  $VO$  for this result set is:  $VO = \{\sigma_{1,t}, \Lambda\}$  and  $\Lambda = \{\Lambda_i | i = 1, 2, \dots, t\}$ .  $\sigma_{1,t}$  is the aggregated signature calculated from the corresponding tuple signatures.  $\Lambda_i$  is the AAI to authenticate record  $i$  in the result set. It consists of all the values of the sibling nodes of those nodes on the path from the selected leaf attribute nodes to the root.

For filtered attributes, we have a definition of **Highest Common Ancestor (HCA)**: Several attributes are said to have a HCA if there exist a subtree in the attr-MHT which has all these attributes and only these attributes as leaf nodes and the root of this subtree is not an ancestor of any other attribute. The root of the subtree is called the HCA of these attributes. In a special case, when a filtered attribute has no common ancestor with all the other filtered attributes, it is said the HCA of this attribute is itself. To illustrate, in Fig. 3  $A_{10}$  is the HCA of itself;  $N_{36}$  is the HCA of  $A_{11}$  and  $A_{12}$ ;  $N_{24}$  is the HCA of  $A_{13}$ ,  $A_{14}$ ,  $A_{15}$  and  $A_{16}$ . Apparently, HCAs are the AAI data needed to be sent to the querier.

To minimize the size of AAI or the number of HCAs, the attr-MHT can be optimized by sorting the attributes according to the attribute query frequency (AQF): the frequency at which an attribute is queried, which can be obtained by query statistics. We give an example to show how to sort the attributes here. Let the most frequently selected attributes come first in the attr-MHT, and the less frequently requested attributes come later. The purpose of this sorting is to produce an ordering of the filtered attributes like  $A_{10}$  to  $A_{16}$  in Fig. 3, to increase the probability that more attributes will have a HCA and thus reduce the total number of HCAs as far as possible.

**$VO$  for SET Operations.** We consider SET operations of UNION and INTERSECT. They are both binary operations which build a new logical relation from two specified relations. A tuple in the logical relation can be mapped into a tuple in either of the two specified relations. The construction of  $VO$  for these operations over two relations uses tuple signatures with relation's name:  $\tilde{\sigma}$ .

UNION is defined as:  $R \cup S := \{t | t \in R \text{ or } t \in S\}$ . It builds a logical relation consisting of all tuples appearing in either or both of two specified relations:  $R$  and  $S$ . For a tuple which belongs to both relations, we make the following rule in choosing which signature for aggregation and verification: if a tuple appears in both relations,

its signature generated by a signer in  $R$  (or  $S$ ) is chosen for signature aggregation; the public key of the signer from  $R$  (or  $S$ ) who signs this tuple is chosen accordingly for signature verification.

Suppose the query result set from a UNIONed operation contains totally  $t$  tuples,  $m$  (inclusive tuples appearing in both relations) from relation  $R$ ,  $n$  (exclusive tuples appearing in both relations) from relation  $S$ , and  $m + n = t$ . Let  $\sigma_i^R$  ( $i = 1, 2, \dots, m$ ) denote the signature of tuple  $i$  from  $R$ ,  $\sigma_j^S$  ( $j = 1, 2, \dots, n$ ) the signature of tuple  $j$  from  $S$ . The  $VO$  for this result set is:  $VO = \{\sigma_{1,t}, \phi\}$  where  $\sigma_{1,t}$  is calculated as  $\sigma_{1,t} = \prod_{i=1}^m \sigma_i^R \times \prod_{j=1}^n \sigma_j^S$ . The querier verifies the result set according to the equation  $e(\sigma_{1,t}, g_2) = \prod_{i=1}^m e(h_i^R, v_i^R) \times \prod_{j=1}^n e(h_j^S, v_j^S)$  where  $h_i^R = H(h_i^R \| \text{“}R\text{”})$  and  $h_j^S = H(h_j^S \| \text{“}S\text{”})$ . The querier is able to verify the result set containing tuples from either or both of the two relations.

INTERSECT is defined as:  $R \cap S := \{t | t \in R \text{ and } t \in S\}$ . The INTERSECT operation builds a logical relation consisting of all tuples appearing in both of two specified relations. The querier needs to verify that every tuple in the result set appears in both relations.

Suppose the query result set from a INTERSECTed operation contains totally  $t$  tuples. Let  $h_i$  ( $i = 1, 2, \dots, t$ ) denote the root value of the attr-MHT for tuple  $i$ . Let  $\sigma_i^R$  denote its signature with  $R$ 's name in  $R$ ,  $\sigma_i^S$  its signature with  $S$ 's name in  $S$ . The  $VO$  for this result set is:  $VO = \{\sigma_{1,t}, \phi\}$  where  $\sigma_{1,t}$  is calculated as  $\sigma_{1,t} = \prod_{i=1}^t (\sigma_i^R \times \sigma_i^S)$ . The querier verifies the result set according to the equation  $e(\sigma_{1,t}, g_2) = \prod_{i=1}^t (e(h_i^R, v_i^R) \times e(h_i^S, v_i^S))$ . The querier is able to verify the result set containing tuples from both of the two relations.

**VO for JOIN.** JOIN is defined as:  $R \bowtie_C S := \{tq | t \in R \text{ and } q \in S \text{ and } C(t, q)\}$  where  $tq$  is a tuple pair,  $C$  is condition of the form  $A_j \Theta A_k$ ,  $A_j$  and  $A_k$  are attributes of relations  $R$  and  $S$  respectively, and  $\Theta \in \{=, \neq, <, >, \leq, \geq\}$ . It builds a relation from two specified relations consisting of all possible concatenated pairs of tuples, one from each of the two specified relations, such that in each pair the two tuples satisfy the specified condition. Without loss of generality, let  $T$  denote the relation resulted from a JOIN  $R \bowtie_C S$ , that is:  $T = R \bowtie_C S = \{p_i | i = 1, 2, \dots, n\} = \{(t_i q_i) | i = 1, 2, \dots, n, t_i \in R, q_i \in S, \text{ and } t_i.A_j \Theta q_i.A_k = TRUE\}$ , where  $p_i$  is a tuple of relation  $T$  and it is in the form of a fixed tuple pair  $(t_i q_i)$ .

The  $VO$  for a JOIN is in the form of  $\{\sigma_{1,n}, \phi\}$ .  $\sigma_{1,n}$  is the aggregated signature by aggregating the signatures of all the  $n$  tuples in relation  $T$  and is calculated as  $\sigma_{1,n} = \prod_{i=1}^n \sigma_i^R \times \prod_{i=1}^n \sigma_i^S$ . The querier uses the equation  $e(\sigma_{1,n}, g_2) = \prod_{i=1}^n e(h_i^R, v_i^R) \times \prod_{i=1}^n e(h_i^S, v_i^S)$  to verify the result set from a JOIN. In case that all the tuples are signed by the same entity,  $v_i^R = v_i^S$ .

However this single step cryptographic signature verification is not enough to authenticate the result set as it cannot detect the reordering attack which is illustrated in Fig. 5. An attacker switches the  $q$  element in the two tuples  $p_1 = (t_1 q_1)$  and  $p_2 = (t_2 q_2)$ . After the switch,  $p_1$  becomes  $p'_1 = (t_1 q_2)$  and  $p_2$  becomes  $p'_2 = (t_2 q_1)$  and the conditions  $C(t_1, q_2)$  and  $C(t_2, q_1)$  may no longer hold. Because  $p_1$  and  $p'_1$ ,  $p_2$  and  $p'_2$  are different in values, they may convey wrong results to the querier and thus en-

danger the querier’s decisions. The reordering attack can occur both on the server side when the server behaves maliciously or is compromised by an outside attacker or during the transmission process when the querier and the server are not communicating over a secure channel, e.g., SSL/TLS. When the server is compromised, the attacker controls the server and reorders the result set before the server sends it out to the querier. When the querier and the server are not communicating over a secure channel, the attacker is able to eavesdrop on the conversation and reorders the result set without destroying it. This reordering attack cannot be detected using only cryptographic verification because the verification of an aggregated signature is order independent and does not check the validity of JOIN condition.

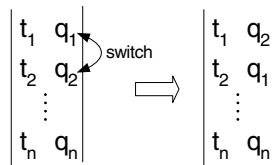


Fig. 5. Reordering attack

To detect reordering attacks, besides the cryptographic signature verification, the querier must also check if the condition  $C(t_i, q_i)$  still holds for each  $p_i$  in the received result set. Thus the authentication process for a JOIN consists of two steps: condition check and signature verification. Before constructing the two attr-MHTs for each tuple  $p_i$  in the result set, the querier checks the condition  $C(t_i, q_i)$  in  $p_i$  first. If the condition  $C(t_i, q_i)$  does not hold the authentication of the result set fails. If the condition check for all the tuples in the result set are successful, the querier further cryptographically verifies the aggregated signature with all the calculated attr-MHT root values. Only when the condition check and the signature verification are both successful can the result set be regarded as authenticated.

#### 4.4 Update Operations

As our attr-MTH is built on individual tuple and each tuple in the table is signed separately and the signature aggregate function is accumulative and communicative, update operations (INSERT, DELETE and UPDATE) can be processed in a direct way: DELETE can be processed by the server with the owner modifying the database’s global information map accordingly; INSERT and UPDATE have to be channelled back to the owner as only the owner possesses the private key for generating new signatures. The owner signs on the new (updated) tuple and sends the new (updated) tuple with the newly generated signature to the server. There is no need to lock the whole table during the updating process. However, to ensure data consistency, the update operations should be done under a concurrency control mechanism like basic 2PL [1].

## 5 Analysis and Evaluation

We analyze our scheme in terms of the following five overheads as defined in [10]: querier communication, querier computation, server computation, server storage and owner computation. The parameters used in the analysis are summarized in Tab. 1:

**Table 1.** Notations

Notation	Meaning	Default
$ S $	Length of a signature (Bytes)	128
$ D $	Length of a hash value (Bytes)	16
$T_R$	Number of tuples/rows in table (million)	1
$T_A$	Number of attributes/columns in table	10
$Q_R$	Number of tuples in the result set	-
$Q_A$	Number of filtered attributes	-
$ A_i $	Size of attribute $A_i$ (Bytes)	-

### 5.1 Querier Communication Overhead

The querier communication overhead consists of two parts: overhead incurred by the aggregated signature and overhead incurred by AAIs. We analyze these two kinds of overheads separately.

**Overhead Incurred by Signature.** The communication overhead incurred by signature remains constant as  $|S|$ , the same as in [10]. It is independent on the number of records in the result set. In contrast, the communication overhead incurred by signature in the MHT-based schemes is linear to  $Q_R$ .

**Overhead Incurred by Authentication Auxiliary Information.** Let  $N_A$  denote the number of digests as AAI per tuple in the result. Suppose the  $Q_A$  filtered attributes are in a continuous sequence in the end of the sorted attribute list.  $N_A$  can be represented as:

$$N_A = \begin{cases} 1, & \text{if } \log_2 Q_A \text{ is an integer;} \\ [2, 2^{\lceil \log_2 Q_A \rceil} + 1], & \text{if } \log_2 Q_A \text{ is not an integer.} \end{cases}$$

When  $\log_2 Q_A$  is an integer, that is  $Q_A = 1, 2, 4, \dots$ , only one digest per record needs to be sent out. When  $\log_2 Q_A$  is not an integer,  $N_A$  is in the range of  $[2, 2^{\lceil \log_2 Q_A \rceil} + 1]$ . Its maximum value  $2^{\lceil \log_2 Q_A \rceil} + 1$  is obtained when  $\log_2(Q_A + 1)$  is an integer. Thus in the worst case when  $\log_2(Q_A + 1) = n$  ( $n$  is an integer), the ratio of  $N_A/Q_A$  is:

$$\frac{N_A}{Q_A} = \frac{2^{\lceil \log_2 Q_A \rceil} + 1}{Q_A} = \frac{2^{\lceil \log_2(2^n - 1) \rceil} + 1}{2^n - 1} = \frac{2^{n-1} - 1}{2^n - 1} < \frac{1}{2} \tag{1}$$

From Eq. 1, we conclude that by employing attr-MHT our scheme reduces the communication overhead incurred by AAI at least by half compared with all the existing authentication schemes.

## 5.2 Querier Computation Cost

The querier computation cost consists of two parts: hashing cost spent on the reconstruction of the attr-MHTs and verifying cost on the aggregated signature.

Hashing cost on the reconstruction of the attr-MHTs is not significant compared to the cost on the aggregated signature verification which is a very expensive operation. Normally, hash functions are about 100 times faster than RSA signature verification [13], more than 1000 times faster than BGLS signature verification. The cost of hashing is dependent on the total length of input bits and is represented as:  $Comp_{hashing} = Q_R * (2 * (T_A - 1) * |D| + \sum_{i=1}^{T_A} |A_i|)$ .

With signature aggregation the querier only verifies one signature to authenticate multiple tuples which is very efficient compared with a naive solution which verifies tuple signatures one by one. To verify a BGLS aggregated signature generated from these  $k \times t$  component signatures, it costs  $k + 1$  bilinear mapping plus  $k \times t - 1$  multiplication operations, that is:  $Comp_Q^{aggregated} = Multi^{k*t-1}(p) + BM(k + 1)$ . To authenticate a query result set from a JOIN, there are additional computation cost incurred by condition check for the querier. However, condition check incurs only a little computation overhead as the comparison operation is very efficiently implemented in modern computers.

## 5.3 Server Computation Cost

Upon a query, besides preparing the result set, the server needs to construct a VO for the result set by aggregating multiple signatures. The server computation cost is calculated as:  $Comp_S = Multi^{Q_R-1}(p) = 0.12 \times (Q_R - 1)$ . The multiplication operation cost involved are not expensive given that  $Multi^1(1) = 0.12ms$  [10], and can be easily mitigated with a powerful server machine. Furthermore the resulting savings in communication overhead and processing cost at the querier side more than justify our proposed scheme.

## 5.4 Server Storage Cost

From the system management angle, although the disk units get cheaper, storage management costs are not getting cheaper and thus the server storage cost is a concern especially in outsourcing models. The attr-MHT incurs no server storage cost. After the signature is calculated, the tree structure which stores all the hash values of the intermediate nodes in the attr-MHT is discarded and only the signature is stored with the physical database. Thus the total cost of server storage of our scheme is calculated as:  $Storage_S = T_R * |S|$ .

## 5.5 Owner Computation Cost

To construct an authentic database, the owner needs to construct an attr-MHT and calculate a signature based on the root value of the attr-MHT for each tuple in the database. The signature generation process can be done off-line. As stated in Section 4.4, because the signature aggregate function is accumulative and communicative, update operations can be processed in a simple and direct way and there is no need to lock the whole table.



## 6 Conclusion

In this paper, we proposed a communication-efficient scheme based on aggregated signature schemes and MHT to authenticate query results disseminated by an untrusted data processing server. To our knowledge, this is the first work that addresses the issue of reducing communication overhead incurred by AAI and the first authentication scheme which supports dynamic JOIN and SET operations.

## References

1. P. Bernstein and N. Goodman. Concurrency control in distributed database systems. In *ACM Computing Surveys*, Volume 13(2), pages 185-221, June 1981.
2. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. in *Advances in Cryptology - EUROCRYPT'2003* (E. Biham, ed.), Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-verlag. Berlin, Germany, 2003
3. C.J. Date. An introduction to database systems (4th Edition). Addison-Wesley, 1985.
4. C.J. Date and H. Darwen. A guide to the SQL Standard (4th Edition). Addison-Wesley, 1997.
5. R. H. Deng, Y. Wu, D. Ma. Securing JPEG2000 Code-Streams. *International Workshop on Advanced Developments in Software and Systems Security*, Dec. 2003
6. P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls and G. Stubblebine. Flexible authentication of XML documents. *Proc. of the 8th ACM conference on Computer and Communication Security*, pp. 136-145, 2001.
7. P. Devanbu, M. Gertz, A. Kwong, and S. Stubblebine. Authentic data publication over the internet. In *14th UFIP 11.3 Working Conference in Database Security*, Pages 102-112, 2002.
8. M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an Authenticated Dictionary with Skip Lists and Commutative Hashing. *Proc. of DISCEX II'01*, Vol. 2, pp. 1068-1083, 2001.
9. R. C. Merkle. A certified digital signature. *Proc. of Advances in Cryptology-Crypto '89*, Lecture Notes on Computer Science, Vol. 0435, pp. 218-238, Springer-Verlag, 1989.
10. E. Mykletun, M. Narasimha, and G. Tsudik, Authentication and integrity in outsourced databases. *NDSS 2004*, Feb. 2004.
11. M. Naor and K. Nissim. Certificate Revocation and Certificate Update. *Proc. of the 7th USENIX Security Symposium*, pp. 217-230, 1999.
12. H.H. Pang, K.L. Tan. Authenticating query results in edge computing. *ICDE 2004*, Mar. 2004.
13. R. Rivest and A. Shamir. PayWord and MicroMint—Two Simple Micropayment Schemes in *Proceedings of 1996 International Workshop on Security Protocols*, (ed. Mark Lomas), Lecture Notes in Computer Science No. 1189, pages 69–87. Springer, 1997.

# Multi-Source Stream Authentication Framework in Case of Composite MPEG-4 Stream

Tieyan Li, Huafei Zhu, and Yongdong Wu

Institute for Infocomm Research ( $I^2R$ ),  
21 Heng Mui Keng Terrace, Singapore 119613  
{litieyan, huafei, wydong}@i2r.a-star.edu.sg

**Abstract.** Multimedia community is moving from monolithic applications to more flexible and scalable integrated solutions. Stream authentication is more complex since a stream may consist of multiple sources and be transcoded by intermediate proxies. In this paper, we propose a multi-source stream authentication (mSSA) framework based on MPEG-4 stream format. We describe the overall authentication architecture and elaborate the encoding, hashing, signing, amortizing and verifying methods used in the basic scheme. Further on, we utilize advanced cryptographic primitives-aggregate signature schemes, to reduce the signatures' size and improve the performance. We illustrate the scheme and discuss the extensions. Our analysis shows that the scheme is secure and efficient.

## 1 Introduction

Multimedia syndication has been put forward for years, for instance, [1] proposed an "InfoPyramid" scheme to interrelate different formatting and conversion options of multimedia objects together with composition strategies for complex multimedia documents. How to protect (*w.r.t.* access control and authentication) these complex media contents is a critical challenge in many applications. The security requirements are different for end roles: on the one hand, media providers want to protect the access to their content; on the other hand, the end users must make sure the authenticity of the content. However, most commercial Multimedia Digital Right Management (DRM) systems, *i.e.* Windows Media Rights Manager (WMRM) for Microsoft [2], are built for the former purpose and ignore the end users' requirement. We hereby focus on authentication of multi-source scalable streams. Moreover, instead of working on meta-data descriptions (*e.g.* specifying usage rules, XrML [3] in DRM), we work directly on specific streaming format-MPEG-4 [4,5] so that standard scalable MPEG-4 stream of multiple sources can be manipulated and verified flexibly.

We give two examples to motivate stream compositions and their respective authentication requirement. First, suppose a local news channel, authorized to broadcast live news from CNN, may change the subtitle from English to its local language. The local channel has to make new commitment on the modified parts so that the combined media stream can be verified by the end users. Thinking

broadly, any tailoring of the original works such as movie advertisement, multi-screening, digital art creation *etc.* are categorized into this class. Second, consider a more complex mix-an interactive video conference, where multiple sources are able to synthesis, distribute and display of customized media content. Similarly, the verification must be robust enough to tolerate various media handlings such as transcoding, filtering, mixing, tilting and switching.

In this paper, we propose mSSA framework for authenticating multi-source scalable stream. Our basic stream authentication scheme, relying on traditional stream authentication mechanisms with signature amortization, enables end-to-end authentication with the transcoding operations. Furthermore, we identify two types of transcoding operations: *truncation-only* with which a proxy is only allowed to truncate a partial sub-stream of an original stream; and *grafting* where a proxy can not only truncate a partial stream, but also insert another sub-stream. In addition, to save bandwidth, we utilize aggregate signature schemes to reduce the size of multiple signatures into one. This first proposed multi-source stream authentication scheme can verify a flexible coded stream in many ways, extend easily and scale well. It can also be adapted into standard DRM systems. Our analysis shows that the scheme is secure and cost-effective.

Paper organization: Section 2 reviews some related works on traditional as well as adaptive stream authentication schemes. We then describe the mSSA framework in section 3. Section 4 introduces the preliminaries, MPEG-4 stream format and depicts the basic authentication mechanisms. In section 5 we elaborate multi-source authentication schemes with type-1 and type-2 transcoding operations. Following on, we analyze the security and performance issues in section 6. At last, we conclude our paper and point out our future tasks.

## 2 Related Works

**Single source stream authentication schemes.** Stream authentication schemes have been intensively studied [10,11,12,13,14,15,16]. The traditional schemes can be categorized as hash graph-based [12], tree-based [11] and symmetric key-based [10]. Other approaches [13,14,15,16] assume an erasure channel, such as the emerging wireless networks-Bluetooth, WPAN, *etc.* where packets are lost from time to time. Erasure codes [9] are then used to tolerate arbitrary patterns of packet loss. However, these works concentrate on end-to-end stream authentication where only single stream source and single receiving end are assumed by default. These traditional stream authentication schemes are failed on Sender-Proxy-Receiver SPR model where an intermediate proxy can manipulate the streaming packets for adapting with fluctuant network conditions. However, these basic works provide us valuable building blocks on stream encoding, packing, hashing, signing and verifying processes.

**Adaptive stream authentication schemes.** Recently, stream authentication in the SPR model was studied in the literature [18,19,20]. [18] proposed a secure MPEG-4 stream authentication scheme that allows a proxy flexibly manipulate streaming packets while they can still be verified by the final receiver. Although

the Unequal Loss Verification (ULV) scheme is secure and efficient, it didn't address the multi-source authentication problem in detail. Suzuki *et al.* in [19] proposed a multimedia content delivery system that protects the end-to-end authenticity of adaptive multimedia. Moreover, the paper used a multi-hop signature scheme for aggregation. Unfortunately, the paper didn't assume an erasure channel where packets are lost arbitrarily. Also, it worked only on meta-data processing, instead of processing streaming content itself. Recently, Gentry *et al.* in [20] proposed two new provably secure schemes, LISSA and TRESSA, that ensure secure streaming media authentication with adaptive proxies. However, it didn't address the multi-source authentication problem. Additionally, it had the problems similar to [19] as it only considered generic multimedia content simply as message blocks instead of real stream format like MPEG-4.

### 3 mSSA Overview

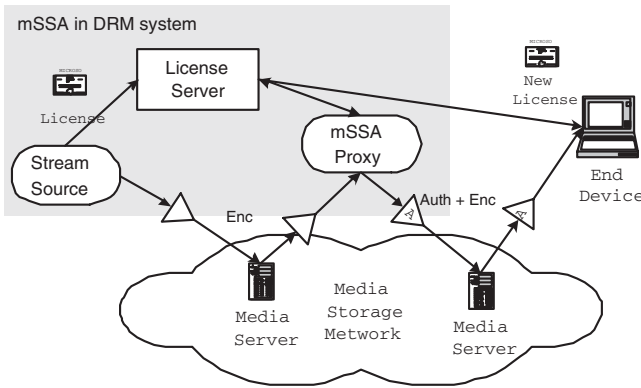
Our basic stream authentication scheme borrows many building blocks from aforementioned traditional stream authentication schemes. Generally, a stream is divided into groups of packets and each group is processed independently. In an erasure channel, the objects/layers of a group at different priority levels are given unequal protection levels via erasure correction coding (ECC) [7]. The various layers are then packed into a group of packets. Furthermore, following the amortizing scheme of SAIDA [13], the producer signs on the hash value of each group, instead of on every packet. The group hash is generated such that recipients are able to verify the source of a stream. It is generally believed that by amortizing the authentication data over a group of packets the verification overheads are much smaller than that of signing on every packet.

Unlike traditional multicast erasure channel where no packet modification is allowed, the proxies in our model are not only a passive packet forwarder, but also have an active role of transcoding and then redistributing the stream into the end network. The transcoding mechanism allows a proxy to discard data layers from the lowest priority layer to higher layers until the resource restrictions are met. For example, Fine Granular Scalability (FGS) [6] is such a scalable mechanism to distribute an MPEG-4 stream efficiently and flexibly over heterogeneous wired and wireless networks. This transcoding strategy differs from packet dropping strategy. Because the transcoded stream can tolerate the same number of packet loss as the original stream, the error-resilience capability is not decreased. Thus, a receiver is able to verify authenticity of the packet origin even if the stream is transcoded. The objective of scalable stream authentication is to authenticate all possible resulting streams after legitimate manipulations on scalable coded streams. We further identify the following two types of transcoding operations (refer to section 5.3 for illustration):

- ◇ Type-1 transcoding operation, *w.r.t.* truncation-only: A proxy is only allowed to truncate one or multiple sub-streams from an original stream. that is to say, there is only one valid stream source.

- ◇ Type-2 transcoding operation, *w.r.t.* grafting: A proxy can selectively truncate one or multiple portions of an original stream, and insert one or more portions from multiple stream sources. All of them form a new stream with multiple sources.

Note that for type-1 transcoding operation, a verifier might verify not only the stream source’s signature, but also the signatures from the proxies for committing their transcoding operations. For type-2 transcoding operation, a verifier has more complex verification procedures involving multiple stream sources and proxies (see section 4 and 5 for details).



**Fig. 1.** Multi-Source Stream Authentication (mSSA) Framework

Fig. 1 sketches the overall mSSA framework. It can be embedded into the current DRM systems as a complimentary media stream authentication scheme. To process stream authentication, the framework consists of three main parts: stream preparation by sources, stream transcoding by mSSA proxies and stream verification by end users. We describe them as follows:

*Part 1: Preparation* The streaming video objects are first encoded according to the MPEG-4 standard. Each source prepares the packets for the object group based on the priorities of the video objects and layers. The source then generates authentication data including integrity units and its signature. The authentication data is amortized over the group of packets. The protected stream as a whole packet group is then ready to be delivered to the proxies<sup>1</sup>.

*Part 2: Transcoding* On receiving the protected stream, a proxy can either apply type-1 transcoding operations to fit the stream into some narrow bandwidth, or employ another protected stream and apply type-2 transcoding operations. In either case, or combined transcoding operations, the proxy needs to sign the new stream with some aggregate signature scheme *AggSigning* (see

<sup>1</sup> The basic authentication scheme is presented in section 4.

section 5). The process can be repeated and finally, the constructed stream is ready to be downloaded by the end users.

*Part 3: Verification* The verification procedure is actually reversing the above two processes. Suppose a receiver retrieves a stream as well as its authentication data from some proxy. It then unpacks, decodes the packets. With recovered authentication data, the receiver can verify the stream integrity and signatures with proper aggregate signature verification scheme `AggVfing`.

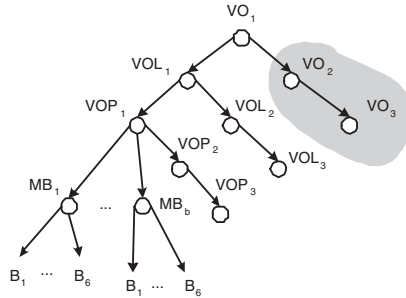
## 4 Basic MPEG-4 Stream Authentication Scheme

### 4.1 Preliminaries and Notations

Notations:  $m$  denotes a message;  $h(\cdot)$  is a collision resistant hash function.  $K_s$  and  $K_p$  denote the private key and public key of the producer; `Sign` is the signing algorithm:  $\sigma = \text{Sign}(K_s, m)$ ; `Vf` is the verification algorithm:  $\text{Vf}(K_p, \sigma, m)$  outputs  $\{true, false\}$ . We also introduce some useful tools like Merkle Hash Tree (MHT) [8] and Erasure Correction Coding (ECC). *Merkle hash tree* has been widely used in many security applications, since it has good security property that it commits on one hash digest over a set of data items. In this paper, we use MHT for generating the integrity data. *Erasure correcting codes* are good means for error resilience of content dissemination in erasure channel. An ECC system with symbols in finite field  $GF(2^w)$  includes two modules: encoding  $Enc_{n,k}(\cdot)$  and decoding  $Dec_{n,k}(\cdot)$ , where  $n$  is the codeword length and  $k$  is the message length.

### 4.2 Syntactic Structure of MPEG-4 Stream

The scheme is based on structure of packets, we first introduce the encoding and packing of an MPEG-4 stream. According to [4,5], an MPEG-4 presentation is divided into sessions including units of aural, visual, or audiovisual content, called media objects. A Video Sequence (denoted as VS, or group) includes a series of Video Objects (VOs). Each VO is encoded into one or more Video Object Layers (VOLs). Each layer includes information corresponding to a given level of temporal and spatial resolution, so that scalable transmission and storage are possible. Each VOL contains a sequence of 2D representations of arbitrary shapes at different time intervals that is referred to as a Video Object Plane (VOP). VOPs are divided further into MacroBlocks (MBs) of size  $16 \times 16$ . Each MB is encoded into six Blocks  $B_1, B_2, \dots, B_6$  of size  $8 \times 8$  when a 4:2:0 format is applied. In an MPEG-4 stream, VOs such as foreground objects and background objects, may have different priorities, indicated as *visual\_object\_priority* taking values  $1 \sim 7$  from lowest to highest priority. In MPEG-4 syntax, each object layer has *visual\_object\_layer\_priority* to represent the importance of different layers. The layer with the highest priority, called the base layer, contains data representing the most important features of the video sequence, while additional layers, called enhancement layers, progressively assigned with lower priorities,



**Fig. 2.** A typical tree structure of an object group with priority levels from  $VO_1, VO_2, \dots$  down to  $VOLs, VOPs, MBs$  and  $Blocks$ . The shadow part that covers subtrees ( $VO_2-VO_3$ ) can be cut off by the transcoding operation.

contain data that further refine the quality of the base layer. The source generates a flow for each layer and assigns to it a unique discarding priority. In Fig.2, we illustrate a typical hierarchical object tree in one visual object group of an MPEG-4 stream.

### 4.3 Generating Authentication Data

The very first step of the basic stream authentication scheme is generating authentication data, which relies on some hashing and signing mechanisms on the packets. Given a group of encoded packets with above tree structure, we are able to generate the hash values bottom-up [18], which is a method similar to the TRESSA hashing scheme in [20], as shown in table 1.

Where  $R$  is the root of the object group and  $G_{ID}$  is the group ID. Thus, the producer can sign the group hash  $h_G$  using its private key  $K_s$  and get the signature as:

**Table 1.** Generating authentication data

<b>Authentication Data Generation</b>
At bottom layer, compute hash values of Blocks: $h_{B_i} = h(Block_i \parallel i), i \in \{1, 2, \dots, 6\}$
Following that, compute hash values of macroblocks: $h_{MB_j} = h(h_{B_1} \parallel h_{B_2} \parallel \dots \parallel h_{B_6})$ <p style="text-align: center;">.....</p>
Upward, suppose an upper layer node $N$ has a set of child nodes $C = (C_1, C_2, \dots, C_c)$ , compute the hash value as: $h_N = h(C_1 \parallel C_2 \parallel \dots \parallel C_c)$ <p style="text-align: center;">.....</p>
At last, we can calculate the group hash as: $h_G = h(R \parallel G_{ID})$

$$\sigma = \text{Sign}(K_s, h_G) \quad (1)$$

By signing once on the root of the object group, the originator actually commits a whole virtual object group to the receivers. Suppose a stream consists of  $n$  virtual object groups,  $n$  signatures are to be generated to authenticate the stream. Hereafter, we use authentication data (denoted as  $\lambda = \langle \sigma, H_G \rangle$ ) to represent both the signatures and the whole or partial<sup>2</sup> group hash values.

#### 4.4 Amortizing Authentication Data

After generating the authentication data, the authentication data is to be amortized into the packets using existing information dispersal algorithm as in [14]. We employ ECC encoders to encode them and amortize them onto the packets before sending them out over an erasure channel (We use the method introduced in [15]). For simplicity, we process the authentication data uniformly with the same encoding rate as of the highest priority layer. For a set of  $n$  packets  $P = \{P_1, P_2, \dots, P_n\}$  and their authentication data  $H_G$  and  $\sigma$  in a  $(k, n)$  erasure channel. The encoding procedure uses the systematic ECC algorithms  $Enc_{2n-k, n}(\cdot)$  and  $Enc_{n, k}(\cdot)$  and computes the codeword  $C_r$  as integrity units  $r_1, r_2, \dots, r_n$ , the codeword  $C_s$  as signature units  $s_1, s_2, \dots, s_n$ , respectively. Next, we append integrity unit  $r_j$  and signature unit  $s_j$  on packet  $P_j$ , for all  $j = 1, 2, \dots, n$ .

#### 4.5 Verifying Authentication Data

The verification process includes unpacking, decoding and verifying, which reverses the generation process. Based on the erasure coding, at least  $k$  out of  $n$  packets of a group should be received in order to recover the authentication data. Suppose  $k$  packets  $\{P_1, P_2, \dots, P_k\}$  are received successfully. The integrity units  $\hat{r}_1, \hat{r}_2, \dots, \hat{r}_k$  and the signature units  $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_k$  are recovered from the received packets. With the decoder  $Dec_{n, k}(\cdot)$  and  $Dec_{2n-k, n}(\cdot)$ , the authentication data is recovered as  $\hat{\lambda} = \{\hat{\sigma}, \hat{H}_G\}$ . Then, the signature can be verified with algorithm  $\text{Vf}(K_p, \hat{\sigma}, \hat{H}_G)$ , where  $K_p$  is the public key of the stream source. If  $\text{Vf}(\cdot)$  is true, then continue to verify the integrity unit; if not, the object group is bogus and discarded. To this end, the end user reconstructs the hash tree  $H'_G$  according to the formulas in table 1 and compares it with the extracted integrity unit  $\hat{H}_G$ . We desire  $H'_G = \hat{H}_G$  for successful verification.

## 5 Multi-Source MPEG-4 Stream Authentication

### 5.1 Basic Transcoding Process

On receiving a stream, a proxy is allowed to do type-1 or type-2 transcoding operations before retransmission. First, we focus on truncate-only transcoding dis-

<sup>2</sup> Depending on how much of a hash tree will be taken as the authentication evidence (the integrity unit), which is the core part of *flexible verification* [18].



cussed in section 3. Based on MPEG-4 stream structure of Fig. 2, truncation-only means that we preserve certain (important) branches of an MHT and truncate other (unimportant) branches to fit the stream into narrow network bandwidth. *I.e.*, the shadow part in Fig. 2 could be truncated if necessary. In this example, we discard the subtree ( $VO_2 - VO_3$ ), retain the hash of the subtree root and keep the subtree ( $VO_1$ ). Apparently, the original authentication data  $\lambda$  has to be changed to a new one  $\lambda'$ . The new data shall contain the original signature  $\sigma$ , the new integrity unit  $H_{VO_1}$  and the new signature  $\sigma_P$  (signed by the proxy on the root of the subtree for committing its modification). We get the new authentication data  $\lambda' = \{\sigma, \sigma_P, H_{VO_1}\}$ . Using above amortization method, we can append them onto the packets and send them out.

Secondly, for grafting transcoding: suppose in above case, the subtree ( $VO_2 - VO_3$ ) is replaced by another subtree ( $T_A$ ) with authentication data  $\lambda_A = \{\sigma_A, H_{T_A}\}$ . Apparently, the authentication data of the composite stream should be  $\lambda'' = \{\sigma, \sigma_P, H_{VO_1}, \sigma_A, H_{T_A}\}$ . And so forth for additional grafting operations (with limitations only by the system capabilities). Same as above,  $\lambda''$  is amortized onto the packets and sent out.

Noted that as the above processes continue, the signature size will be linearly increased and proportional to the signing parties. The overheads of these signatures will soon become unaffordable after several transcoding operations. Fortunately, there are two methods to reduce the size of the authentication data. Firstly, we can reduce the size of integrity unit by dropping some hash values of the subtrees. Suppose we generate a hash tree over totally  $n$  leaf nodes. If we cut off all hash values of those leaf nodes, we remain  $n - 1$  hash values. Bottom up, if we cut off lower  $m$  levels of the tree, we have only  $n/2^{m-1} - 1$  hash values. However, the tradeoff of using this method is that it can not refine the verification. Thus, we hereby concentrate on reducing the signature size that do not loss any verification granularity as well as security. We employ a cryptographic primitive, aggregate signature schemes such as [21][22], to significantly reduce the size of multiple signatures into one only.

## 5.2 Transcoding with AggSigning and AggVfing

One special variant—a sequential signature scheme [22]<sup>3</sup> can be applied into our type-1 transcoding operation for reducing the signature size. Here, the proxy  $P_i$  takes as inputs the signature  $\sigma_{i-1}$  from its predecessor and all integrity data  $\Sigma_i$  as a whole, and outputs aggregate signature  $\sigma_i$ . Thus,  $\lambda_i = \{\sigma_i, \Sigma_i\}$ . The new authentication data is amortized onto the packets and sent to the end users. Suppose an end user received at least  $k$  out of  $n$  packets of a stream in order to recover the authentication data. The integrity unit  $\hat{\Sigma}_i$  and the signature unit  $\hat{\sigma}_i$  are recovered from the received packets. Then, the signature can be verified with above algorithm  $\text{AggVf}(\hat{\sigma}_i, \hat{\Sigma}_i)$  with corresponding public keys  $\{PK_1, \dots, PK_i\}$ . If  $\text{AggVf}(\cdot)$  is true, then continue to verify the integrity unit; if not, the object group is bogus and discarded. The end user reconstructs the hash tree  $\Sigma'_i$

<sup>3</sup> Since the scheme is built on standard primitives like RSA, that is widely adopted.

and compares it with  $\hat{\Sigma}_i$ . If  $\Sigma'_i = \hat{\Sigma}_i$ , it means successful verification. If not, it indicates a mismatch between two hash trees.

### 5.3 Illustration

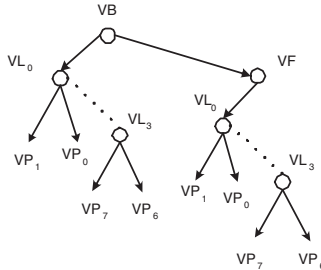
In this section we illustrate above transcoding concepts with an encoded MPEG-4 stream. Shown in Fig. 3, the images are generated artificially with English characters as foreground objects. The background objects are considered important and are coded with  $Enc_{5,4}(\cdot)$  ECC, but the foreground objects correspond to layers of lower priority and are not coded for error resilience. We draw some assumptions to simplify the demo: 1, we ignore the experiments for packet loss; 2, image pixels are used instead of DCT coefficients for data units in the process of hash computation which could be sufficient for demonstrating our scheme; 3, each layer includes two contiguous bit planes; and 4, we consider only two critical concepts, VO and VOP, in MPEG-4 visual objects. The syntactic structure of the simplified image sequence is shown in Fig. 4.

In order to allow authentication of the transcoded stream, for type-1 transcoding, the proxy generates the hash for the covering-subtree value  $HVF$  and incorporates one patch into each packet. For type-2 transcoding, the proxy cuts and pastes a subtree  $VF'$  replacing  $VF$ . Figure 5 and 6 illustrate the syntactic structure of type-1 and type-2 transcoded objects, respectively. Then, the proxy sends the modified packets to the clients.

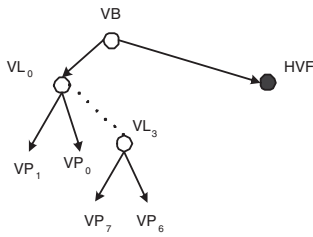
In type-1 transcoding, *e.g.*, the MPEG-4 stream is adapted to a narrow bandwidth wireless network, the proxy filters out the foreground objects and



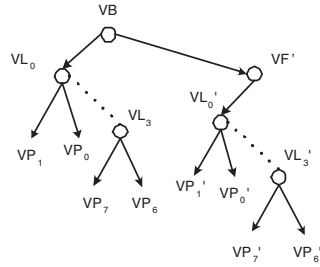
**Fig. 3.** Sample image sequence. The background images forms the basic layer. While the foreground objects (English words) form a sentence.



**Fig. 4.** Syntactic structure of image objects. Where VB denotes the background object, and VF denotes the foreground object.



**Fig. 5.** Syntactic structure of type-1 transcoded objects



**Fig. 6.** Syntactic structure of type-2 transcoded objects



**Fig. 7.** Received sample image sequence with transcoded foreground objects

just transmits the background objects to clients. In type-2 transcoding, *e.g.*, the proxy changes the foreground objects with some advertisements. Fig. 7 illustrates the image sequence received by the clients for type-2 transcoding.

### 5.4 Discussions

Above we illustrate how a portion of a stream can be replaced with a portion of another stream. In fact, multi-source stream can be flexibly composed. For instance, in case of multi-screening, a super composer can summarize multiple streams in a big screen, where each stream is scaled down to fit into its small screen. To sign such a super stream is straightforward, the composer may aggregate the signatures attached with those streams and generate its aggregate signature. However, if a portion of a stream is grafted as in Fig. 6, *i.e.* suppose  $VF'$  is a subtree rooted at  $VB'$  where only  $VB'$  is signed but not  $VF'$ <sup>4</sup>, then where can we sign? Fortunately, the scheme can be extended easily by providing proof of trust from the signed root to the replacing portion. *I.e.* we need to prove  $VF'$  is indeed a subtree of  $VB'$  by taking as evidence the corresponding hash values. Another way is to sign on every possible portions of the stream so that we take that portion directly and generate the aggregate signature. Both methods may introduce some overheads.

One critical concern is whether the scheme can be embedded into the legacy DRM systems, as mentioned in section 3. First, let's review the process of scalable multimedia. Raw multimedia stream is compressed once with a scalable

<sup>4</sup> In our scheme, each stream is signed once on the root implicitly.

coding scheme and the resulting codestream can be decoded adaptively. The protection strategies must conform with the scalability of the codestream, thus the fundamental property of (generally upper layer's) authentication and encryption is to preserve the scalability. Current DRM systems are focusing on protecting illegal access to the content. End users download content from content distributors and separately, obtain the decryption keys from a license server. Authentication is applied after the coding phase and normally before the encryption process. Here, authentication/verification and encryption/decryption is two separate processes since they are based on different (Asymmetric/symmetric cipher) techniques. In a shared key case, it is also possible to use authenticated encryption schemes such as OCB [23]<sup>5</sup>.

## 6 Security and Performance Analysis

The security of our scheme relies on the security of the Merkle hash tree and the aggregate signature scheme. Fortunately, Merkle hash tree has very nice security properties [8] and the security of aggregate signature is analyzed in [21,22]. Thus, we concentrate on the integrity unit that is how much percentage of a stream is verifiable. Then, we analyze the computational cost of each party in the scheme.

### 6.1 Percentage of Verification

Assuming the authentication data is recoverable as it has the highest priority as the base layer. Additionally, assuming an erasure channel with independent packet losses, given  $\rho$  the packet loss probability. The group of  $n$  packets transferred over the erasure channel may have probably  $\binom{n}{k}\rho^{n-k}(1-\rho)^k$  packets received. The verification delay for a group of  $n$  packets is  $O(n)$ . In our definition, only those recovered content of a received stream can be verified. From the recovered authentication data  $\hat{\lambda}_i$  and  $\text{AggVf}(\hat{\sigma}_i, \hat{\Sigma}_i)$ . We know the validity of the signatures. Further on, if we don't receive enough packets to recover all stream, we are not able to verify the full integrity unit  $\hat{\Sigma}_i$  from the reconstructed hash tree  $\Sigma'_i$ . Let  $P = \Sigma'_i / \hat{\Sigma}_i$  denote the percentage of verification. We claim that the rate of the reconstructed hash tree  $\Sigma'_i$  over the recovered hash tree  $\hat{\Sigma}_i$  from the received packets directly determines the verification rate  $P$  over the object group, given the signature on  $\hat{\Sigma}_i$  is valid.

### 6.2 Computational Cost

We assume that the computational cost relates to security operations without encoding/decoding costs. Additionally, the computational cost for signature generation and verification depends on the signature scheme selected and normally, the signature verification is considered much faster than signature generation.

---

<sup>5</sup> This will be a totally different scheme and not discussed further due to space limitation.

And, for an MHT with  $n$  leaf items, the total number of hash operations is roughly  $2n$ .

For type-1 transcoding, there is only one stream source with one signature generation and  $2n$  hashing operations. Each intermediate proxy  $P_i$  must first verify  $i - 1$  former signatures and generate one sequential aggregate signature for all its transcoding operations. The final receiver has to verify all the signatures of the aggregate signature, but at relatively lower cost. On the other hand, the receiver will also spend time on reconstructing the (probably partial) hash tree over the object group. For type-2 transcoding, each stream source  $S_i$  generate one signature and  $2n_i$  hashing operations. Each intermediate proxy  $P_i$  must first verify  $i - 1$  former signatures and generate one general aggregate signature for all its type-2 transcoding operations. The cost at the final receiver is the same as that of type-1.

## 7 Conclusions and Future Works

We proposed the first secure multi-source authentication scheme for composite MPEG-4 stream. The scheme works under the general assumption of “erasure channel”, but can be adapted to “polluted erasure channel”, *e.g.* by using distillation code [17]. We elaborated how the encoding, hashing, transcoding, signing and verifying mechanisms are integrated in the mSSA framework. The scheme extends easily and scales well. Our analysis shows that it is secure and cost-effective. The detailed scheme can complement to some standard DRM platforms to protect multimedia content. In the futue, we will develop the innovative prototype within a legacy DRM framework.

## References

1. J.R. Smith, R. Mohan, C. S. Li, *Scalable Multimedia Delivery for Pervasive Computing*. In Proc. ACM Intl. Conf. on Multimedia (ACMM'99), Orlando, FL, 1999.
2. Microsoft, *Architecture of Windows Media Rights Manager*, <http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx>
3. eXtensible right Markup Language (XrML). <http://www.xrml.org>
4. ISO/IEC 14496-1:2001 Information Technology-Coding of Audio-Visual Objects-Part 1: Systems.
5. ISO/IEC 14496-2:2003 Information Technology-Coding of Audio-Visual Objects-Part 2: Visual.
6. Weiping Li, *Overview of fine granularity scalability in MPEG-4 video standard*, IEEE Trans. on Circuits and Systems for Video Technology, 11(3):301-317, 2001.
7. A. E. Mohr, E. A. Riskin, R.E Ladner, *Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction*. IEEE Journal on Selected Areas in Communications, 18(6):819-828, 2000.
8. R. C. Merkle, *A certified digital signature*, Crypto'89, Lecture Notes on Computer Science, Vol. 0435, pp. 218-238, Spriner-Verlag, 1989.
9. M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, *Practical loss-resilient codes*, in Proc. 29th Annual ACM Symposium on Theory of Computing (STOC'97), El Paso, TX, May 1997.

10. A. Perrig, R. Canetti, J. D. Tygar, and D. Song. *Efficient authentication and signature of multicast streams over lossy channels*. In Proceedings of the IEEE Symposium on Research in Security and Privacy (S&P'00), pages 56-73, May 2000.
11. P. Golle and N. Modadugu, *Authenticated streamed data in the presence of random packet loss*, in Proc. Network and Distributed System Security Symposium (NDSS'01), San Diego, CA, Feb. 2001.
12. S. Miner and J. Staddon. *Graph-based authentication of digital streams*. In Proceedings of the IEEE Symposium on Research in Security and Privacy (S&P'01), pages 232-246, May 2001.
13. J. M. Park, E. K. Chong, and H. J. Siegel. *Efficient multicast packet authentication using signature amortization*. In Proceedings of the IEEE Symposium on Research in Security and Privacy (S&P'02), pages 227-240, May 2002.
14. J. M. Park, E. Chong, and H. J. Siegel. *Efficient multicast packet authentication using erasure codes*. ACM Transactions on Information and System Security (TIS-SEC'03), 6(2):258-285, May 2003.
15. A. Pannetrat and R. Molva, *Efficient multicast packet authentication*, in Proc. Network and Distributed System Security Symposium (NDSS'03), San Diego, CA, Feb. 2003.
16. Maxwell N. Krohn, Michael J. Freedman, David Mazires *On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution*. IEEE Symposium on Security and Privacy (S&P'04), California, USA.
17. C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, *Distillation codes and applications to DoS resistant multicast authentication*, in Proc. 11th Network and Distributed Systems Security Symposium (NDSS'04), San Diego, CA, Feb. 2004.
18. Teyan Li, Yongdong Wu, Di Ma, Huafei Zhu, Robert H. Deng, *Flexible Verification of MPEG-4 Stream in Peer-to-Peer CDN*, 6th International Conference on Information and Communications Security (ICICS'04), LNCS 3269, Spain, 2004.
19. Takashi Suzuki, et al. *A system for end-to-end authentication of adaptive multimedia content*. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS'04), Sept. 2004.
20. C. Gentry, A. Hevia, R. Jain, T. Kawahara, and Z. Ramzan. *End-to-End Security in the Presence of Intelligent Data Adapting Proxies: the Case of Authenticating Transcoded Streaming Media*. J. Selected Areas of Communication, Q1, 2005.
21. Dan Boneh, Craig Gentry, Ben Lynn, Hovav Shacham. *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*. EUROCRYPT 2003.
22. Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, Hovav Shacham. *Sequential Aggregate Signatures from trapdoor one-way permutations*. EUROCRYPT 2004.
23. P. Rogaway, M. Bellare, J. Black and T. Krovetz, *OCB: A block cipher mode of operation for efficient authenticated encryption*, in Proc. of 8th ACM CCS'01.

# Batching SSL/TLS Handshake Improved\*

Fang Qi<sup>1,2</sup>, Weijia Jia<sup>1</sup>, Feng Bao<sup>2</sup>, and Yongdong Wu<sup>2</sup>

<sup>1</sup> School of Information Science and Engineering,  
Central South University, Changsha 410083, China  
csqifang@mail.csu.edu.cn, itjia@cityu.edu.hk

<sup>2</sup> Institute for Infocomm Research,  
21, Heng Mui Keng Terrace, Singapore, 119613  
{stufq, baofeng, wydong}@i2r.a-star.edu.sg

**Abstract.** Secure socket layer (SSL) is the most popular protocol to secure Internet communications. Since SSL handshake requires a large amount of computational resource, batch RSA was proposed to speedup SSL session initialization. However, the batch method is impractical since it requires a multiple of certificates. In this paper, we overcome this problem without modifying SSL protocol. To select the optimal batching parameters in terms of performance of server and durable waiting time of the client, we model the connection request with M/D/1 queue. We validate the solutions of the analytical model through simulation.

## 1 Introduction

Secure communication such as Internet banking and e-commerce [1] is an intrinsic demand of today's world of online transactions. As the most widely used method, SSL/TLS [2] runs above existing protocols like TCP. It protects communications by encrypting messages with a secret key negotiated in the SSL handshake protocol [3]. The SSL Handshake Protocol allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before transmitting and receiving the first byte of data [4]. However, SSL handshake protocol needs intensive computational resource due to the cost of public-key operations. For example, a typical Pentium server (running Linux and Apache) can handle about 322 HTTP connections per second at full capacity but only 24 SSL connections per second; and a Sun 450 (running Solaris and Apache) fell from 500 to 3 connections per second [5]. To improve the performance of SSL/TLS handshake protocol, there are several ways:

- (1) *Hardware*: Obviously, a specific circuit can improve the performance. This solution may not be a good solution to middle or small servers [6].
- (2) *Session Caching*: the cache allows subsequent connections to resume an earlier TLS session and thus reuse the result of an earlier computation. Research

---

\* The first author's work is done during her attachment to Institute for Infocomm Research under its sponsorship. This effort is partially sponsored by the National Basic Research Program (973) MOST of China under Grant No. 2003CB317003.

has suggested that, indeed, session caching helps web server performance [7]. However, the cache technology has no help to speedup the session setup.

- (3) *Batching*: Fiat [8] presented an algorithmic approach for speeding up SSL’s performance on a web server by batching the SSL handshake protocol. It is designed for heavily-loaded web servers handling many concurrent SSL sessions. Shacham and Boneh [9] improved the batching performance with batching multiplication-inversion.

In this paper, we focus on the batching technology to improve the performance of SSL. Shacham and Boneh [9] prove that it is impossible to use a single certificate in the present SSL/TLS system. Whereas, we adapt the certificate mechanism so as to provide SSL/TLS setup with only one certificate issued by Certificate Authority.

We also model the client request as a M/D/1 queue and use its approximate solution to optimize the batch size of the server. This optimal batch size satisfies the client’s requirement for the stability of the batching system and enables the mean response time of a batch system behaves nicely using the the optimal batch size.

The rest of the paper is organized as follows. To be self-contained, an overview of the batch RSA [8] is presented in Section 2. The unique certificate schemes in batch SSL are presented in Section 3. We propose the method to select optimal batching parameters in terms of client convenience and server computational cost in detail in Section 4. Finally, we validate the solutions of the analytical model through simulation in Section 5.

## 2 Review of Batch RSA

As a SSL server waits for more than one RSA decryption request and performs one big computation for all decryptions, it can spare a lot of running time capacity [8], being able to perform more SSL handshakes.

Given  $b$  distinct and pairwise relatively prime public keys  $e_1, \dots, e_b$ , all sharing a common modulus  $N = pq$ .  $n$  is the bit length of the public modulus  $N$  and  $k$  the bit length of the bigger of  $e_i$ . Furthermore, we have  $b$  encrypted messages  $v_1, \dots, v_b$ , one encrypted with each key, which we wish to decrypt simultaneously, to obtain the plaintexts  $m_i = v_i^{1/e_i}$ .

In the following equations, quantities subscripted by  $L$  or  $R$  refer to the corresponding value of the left or right child of the node, respectively. Fiat’s algorithm consists of three phases: a multiplication computation phase, exponentiation phase and division computation phase. In the multiplicative product computation phase, we seek to combine the individual encrypted messages  $v_i$  to form, at the root of the batch tree, the value  $v = \prod_{i=1}^b v_i^{e/e_i}$ , where  $e = \prod_{i=1}^b e_i$ .

Using the binary tree construction, working from the leaves to the root. At every internal node, each encrypted message  $v_i$  is placed (as  $v$ ) in the leaf node labeled with its corresponding  $e_i$ . The  $v$ ’s are percolated up the tree using the following recursive step, applied at each inner node:

$$v \leftarrow v_L^{E_R} \cdot v_R^{E_L} \tag{1}$$



$E_L, E_R$  are the left child and right child of each product of internal node.

At the completion of the multiplication computation phase, the root node contains  $v = \prod_{i=1}^b v_i^{e/e_i}$ . In the exponentiation phase, the  $e$ th root of this  $v$  is extracted. The exponentiation yields  $v^{1/e} = \prod_{i=1}^b v_i^{1/e_i}$ , which we store as  $m$  in the root node.

In the division computation phase, each plaintext can be recovered with the following formula:

$$M_i = \frac{v}{v_i^{(a_i-1)/e_i} \prod_{j=1, j \neq i}^b v_j^{a_i/e_j}} \pmod N \tag{2}$$

where  $a_i$  is calculated using the theory of Chinese remainder theorem.

This batching technique is only worthwhile when the public exponents  $e_i$  are very small (e. g. , 3 and 5). Otherwise, the extra arithmetic required is too expensive. Also, notice that one can only batch-decrypt ciphertexts encrypted using distinct public exponents. This is essential. Indeed, Shacham and Boneh [9] showed that it is not possible to batch when the same public key is used.

### 3 Unique Certificate Scheme in Batch RSA

In the standard SSL protocol, each client encrypts a 48-byte pre-master secret using  $e_i$  as the encryption exponents, and the server decrypts the ciphertext independently so as to get the pre-master secret. But batch RSA obtains the pre-master secrets from a multiple clients and hence improve the performance significantly. Unfortunately, batching schemes [8][9] has following disadvantages: requirement for many different RSA certificates; additional payment for certificates; extra maintenance works of multiple certifications. There are two ways to solve this multiple certificates problem in existing [8][9].

- (1) The server issues sub-certificates for himself as sub-Certificate Authority. The public keys of sub-Certificate which is sent to the clients are the encryption exponents  $e_i$  using in batch RSA. This method has to ask the client to check one more certificate.
- (2) Another method is to re-use the message `serverHello.random` in the protocol as shown in Table 1. For simplicity, We only show the related processes and the modified information in the standard SSL/TLS handshake protocol. In this improvement,  $e_i$  is actually a part of `serverHello.random`. Server only need send unique certificate to all the clients. This solution requires no modification on SSL protocol because the encryption exponents  $e_i$  is assigned to the client in `serverHello.random`. The client will verify the certificate as usual, but encrypt the pre-master secret with received  $e_i$  instead of the public exponent in the certificate. Therefore, no extra charge is required, and it is easy to manage the certificate. On the other hand, since the certificate is used to prove the owner who knows the factors of the RSA moduli  $N$  only, this adaption does not undermine the security strength of SSL protocol.

**Table 1.** Unique Certificate for a partial handshake

Client 1	Server
ClientHello	→
	← ServerHello.random including $e_1$
	← Certificate*( $N, e$ )
Certificate*	
ClientKeyExchange	→
( $v_1 = m_1^{e_1}$ )	

Client 2	Server
ClientHello	→
	← ServerHello.random including $e_2$
	← Certificate*( $N, e$ )
Certificate*	
ClientKeyExchange	→
( $v_2 = m_2^{e_2}$ )	

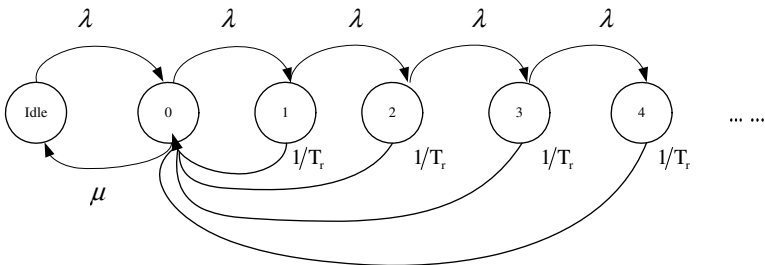
## 4 The Optimization of Batching Parameter

In this section, we propose a method to select optimal batching parameters in terms of client convenience and server computational cost.

### 4.1 Batching Queue Model M/D/1

Suppose the client arrival process is Poisson distributed with an arrive rate  $\lambda$ , and the server response is regarded as an M/D/1 queue. Let the batching service time be  $\tau$  which is determined by the batching size  $b$ .

The M/D/1 model can be approximated with a semi-Markov process depicted in Figure 1. The state of semi-Markov process is described by  $(i)$  where  $i$  indicates the number of clients waiting in the queue with  $i$ =idle indicates the server is idle. Let the mean residual service time is  $T_r = 0.5\tau$  [10].



**Fig. 1.** Semi-Markov Process Model of Batching server,  $\mu = 1/\tau$

Let  $\psi_0$  be the mean holding time in state idle,  $\psi_1$  be the mean holding time in state (0), and  $\psi_2$  be the mean holding time in state ( $i$ ), for  $i > 0$ . Thus

$$\begin{aligned} \psi_0 &= \int_0^\infty t\lambda e^{-\lambda t} dt = \frac{1}{\lambda} \\ \psi_1 &= \int_0^\tau t\lambda e^{-\lambda t} dt + \int_\tau^\infty \lambda e^{-\lambda t} dt = \frac{1}{\lambda}\eta_1 \\ \psi_2 &= \int_0^{T_r} t\lambda e^{-\lambda t} dt + \int_{T_r}^\infty \lambda e^{-\lambda t} dt = \frac{1}{\lambda}\eta_2. \end{aligned}$$

where  $\eta_1 = 1 - e^{-\lambda\tau}$  denotes the Markov transition Probability from state idle to state(0), and  $\eta_2 = 1 - e^{-0.5\lambda\tau}$  denotes the Markov transition Probability from state( $i$ ) to state( $i+1$ ), for  $i > 0$ .

Then, we can solve the steady distribution  $\pi_0^*$  for semi-Markov due to [11] as

$$\pi_0^* = \frac{(1 - \eta_1)(1 - \eta_2)}{1 + \eta_1\eta_2 - \eta_2} \tag{3}$$

Then we compute the queueing waiting time  $T_q$  using the residual service time  $T_r$  and the steady state distribution  $\pi_0^*$  for semi-Markov when the server is idle.

$$T_q = (1 - prob(idle)) \times T_r = (1 - \pi_0^*) \times T_r \tag{4}$$

### 4.2 Optimal Parameter

Clearly, the batching size  $b$  is related to the quality of server service. Thus, we need to select an optimal parameter  $b$  for optimizing the performance of server and clients.

Let the batching RSA decryption time in SSL handshake time is  $T_B$ . Since  $T_B$  is the majority of service time, the batching service time of the server is roughly  $T_B$ .

**Theorem 1:** *To satisfy the client’s requirement for the stability of the system, the batching service time is less than the batch size multiply poisson distributed mean arrival time interval when the time  $t \rightarrow \infty$  in the Batch Queue Model M/D/1, thus*

$$\tau \approx T_B < \frac{b}{\lambda} \tag{5}$$

**Proof:** Let  $\{X_i\}$  ( $i = 1, 2, \dots$ ) be the arrival time interval of two consecutive requests, and  $Y$  be the time interval of  $b$  consecutive requests. According to [10], if the system achieves the stability when the time  $t \rightarrow \infty$  for M/D/1 queue model,

$$T_B < E(Y)$$

where  $E(Y)$  is the expected value of  $Y$ . Because the  $X_i$  are random variable with independent identical distribution, the average arrival time interval of  $b$  consecutive requests is

$$E(Y) = E\left(\sum_{i=1}^b X_i\right) = bE(X_i) = b/\lambda \tag{6}$$

Then the theorem 1 is proved. □

**Lemma 1:** *In the Batch Queue Model M/D/1, to satisfy the client’s requirement for the stability of the system, thus*

$$T_q < \frac{b}{2\lambda} \tag{7}$$

**Proof:** In the Batch Queue Model M/D/1,we first continue deriving the value of  $T_q$  following the Eq 4

$$\begin{aligned} T_q &= (1 - prob(idle))T_r = (1 - \pi_0^*)T_r \\ &= \left(1 - \frac{(1-\eta_1)(1-\eta_2)}{1+\eta_1\eta_2-\eta_2}\right)T_r = \frac{1-e^{-\lambda\tau}}{1-e^{-\lambda\tau}+e^{-1.5\lambda\tau}}T_r \\ &= \frac{e^{\lambda\tau}-1}{e^{-\lambda\tau}-1+e^{-0.5\lambda\tau}}T_r = \frac{1}{1+\frac{1}{(e^{\lambda\tau}-1)e^{0.5\lambda\tau}}}T_r \\ &< \frac{0.5\tau}{1+\frac{1}{(e^{\lambda\tau}-1)e^{0.5\lambda\tau}}} \end{aligned} \tag{8}$$

where  $T_r = 0.5\tau < \frac{b}{2\lambda}$ .

Due to Theorem 1  $\lambda\tau < b$ , it is easily described as

$$T_q < \frac{1}{1 + \frac{1}{(e^b-1)e^{0.5b}}} \tag{9}$$

Especially, when  $b = 2$ ,  $\frac{1}{1 + \frac{1}{(e^2-1)e^{0.5b}}} \approx 0.944$ . The value of this formula is increased with the  $b$ , while not exceed 1. We can get the value bound of the upper limit of  $T_q$  as  $[0.944\frac{b}{2\lambda}, \frac{b}{2\lambda})$ . Then the Lemma 1 is proved. □

From the point of view of the client, the total mean responding time  $T = T_q + T_c + T_B$ , where  $T_c$  is the time for waiting other client in the same batching. We can easily derive the max value of  $T_c$  is  $(b - 1)/\lambda$  while the arrive rate is  $\lambda$ . Otherwise, the upper limit of  $T_b$  has been estimated as  $\frac{b}{\lambda}$  in Theorem 1.

It is supposed that the total client response time dose not exceed the client’s psychologically durable waiting time  $T_d$ . Then, we can construct the following inequation to estimate the approximate bound of batching size.

$$T = T_q + T_c + T_B < \frac{b}{2\lambda} + \frac{(b-1)}{\lambda} + \frac{b}{\lambda} < T_d \implies b < \frac{2}{5}(\lambda T_d + 1) \tag{10}$$

In summary, to satisfy the customer’s requirement for the stability of the system,we can get the optimal solution which satisfy  $T_B < b/\lambda$  derived from theorem 1. On the other hand, it is supposed that the solution of  $b$  should

satisfy the approximate bound described as Eq 10. which assumes that the total customer response time will not exceed the customer’s psychologically durable waiting time. The optimal parameter of batch size can be solved by the following inequations. The reason to select  $b$  as large as possible is to reduce the number of decryption request to decryption device.

$$\begin{cases} 2 \leq b \leq \lfloor 0.4(\lambda T_d + 1) \rfloor \\ T_B < b/\lambda \end{cases} \tag{11}$$

**Determining  $T_d$ :** If the client’s durable waiting time is known, we can estimate the upper limit of batch size  $b$  according as different arrival rate of the client. We further consider the constants and practical value of the  $T_d$  to estimate the upper limit of the batch size  $b$ . On disconnected thinking, 50% people apparently don’t look beyond the first page of search results, and will wait only 8 seconds for a website to download before they get fed up and move on [12]. So we assume durable waiting time  $T_d \approx 8$ . That is to say, if the response time  $T_d$  exceeds 8 seconds, the client will cancel the request.

**Estimating  $T_B$ :** Let  $n$  be the bit length of the public modulus  $N$  and  $k$  be the bit length of the bigger of exponent  $e_i$ . Because the public exponents  $e_i$  are chosen as small as possible to make auxiliary exponentiations cheap, the low-cost operations in the computation cost estimation can be ignored.

The performance analysis of the batch RSA exponentiation algorithm can be divided as multiplication, exponentiation and division computation phases as discussed in Section 2.

We can estimate the cost of  $e = \prod_{i=1}^b e_i$  is  $(b - 1)n^2$ . The cost of computing  $1/e$  is  $(b - 1)n^2$ . The main computation cost is exponentiation costs  $(3k - 2)n^2 + o(n^2)$  with the exponent  $e_i$  where the bit length of  $e_i$  is  $k$ . The whole cost in multiplication phase is  $b(2 + 3k)n^2 + o(n^2)$ .

In the exponentiation computation phase  $v^{1/e} \pmod N = \prod_{i=1}^b v_i^{1/e_i} \pmod N$  costs  $3n^3 + n^2 + o(n^2)$ .

The division computation phase use the theory of Chinese remainder theorem. The phase requires a total of 4 small exponentiations, 2 inversions, and 4 multiplications at each of the  $(b - 1)$  inner nodes.

Roughly, the cost for two inversion operations is equivalent to that of 40 multiplications of  $n$ -bit integers. Thus, the whole cost in the division phase is almost  $(40b + 3kb^3 - 1)n^2 + o(n^2)$ .

As a result, the batching RSA decryption time is

$$T_B = \left( \frac{3n^3 + n^2(42b + k(3b^3 + 3b) - 1)}{b(3n^3 + n^2)} \right) bT_{rsa} = \left( \frac{3n + 42b + k(3b^3 + 3b) - 1}{b(3n + 1)} \right) bT_{rsa}$$

## 5 Validation of Analytical Optimal Parameter

In this section we validate our analysis by comparing the analytical results obtained above with the simulation results.

### 5.1 Experiment Configuration

We perform the simulation by executing conventional RSA and batch RSA using the Cryptix [14] implementation. The time of the conventional RSA and batch RSA decryption and the other parameters such as the mean responding time  $T$ , mean queueing waiting time  $T_q$ , the time waiting other client in the same batching  $T_c$  and batching service time  $T_B$  of the analytical models are executed on a machine with an Dell Intel Pentium IV processor clocked at 3. 20GHz and 1GMB RAM.

Specifically, we perform the simulation of batching RSA with very small public exponents, namely  $e = 3, 5, 7, 11, etc.$  The simulation result of the conventional RSA decryption time  $T_{rsa}$  with larger public exponent, namely  $e = 65537$  is about 32ms which is tested using reiterative results. Both algorithms assume public modulus  $N$  is 1024 bits length. Otherwise, because SSL protocol usually uses a random 48-bytes string  $R$  as Pre-MasterSecret, we assume the length of plaintext is 384bits.

### 5.2 Experiment Results

**Optimal Batch Size Validation:** Figure 2 validates our approximation of optimal parameter of batch size  $b$  described by the inequations 11.

For relative small arrival rates,  $b$  is almost uniform calculated by our analytical model, especially for arrival rates  $\lambda < 2$  as shown in figure 2. Since arrival rates are small(i. e. ,  $\lambda < 0.6$ ),there is very little opportunity to batch, and therefore, the solution of  $b$  are relative small as shown in figure 2. Even at higher arrival rate, the analytical result and simulation result are very close. As shown in figure 2, the solution of the optimal batch size are increased with  $\lambda$  both in analytic and simulation when  $\lambda < 4$  approximately. When  $4 < \lambda < 30$ ,  $b$  is 13 and the optimal batch size reaches Maximum. Although, the speed of batch RSA with  $b = 13$  compared with  $b = 4$  proposed by Shacham and Boneh [9] (thereafter we call this SB scheme) is not very fast, we effectively reduce the number of

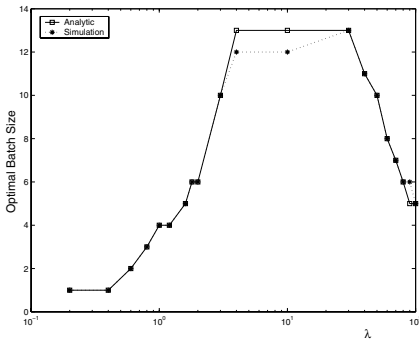


Fig. 2. Optimal Batch Size Validate

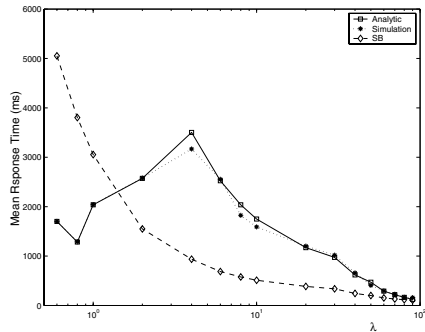
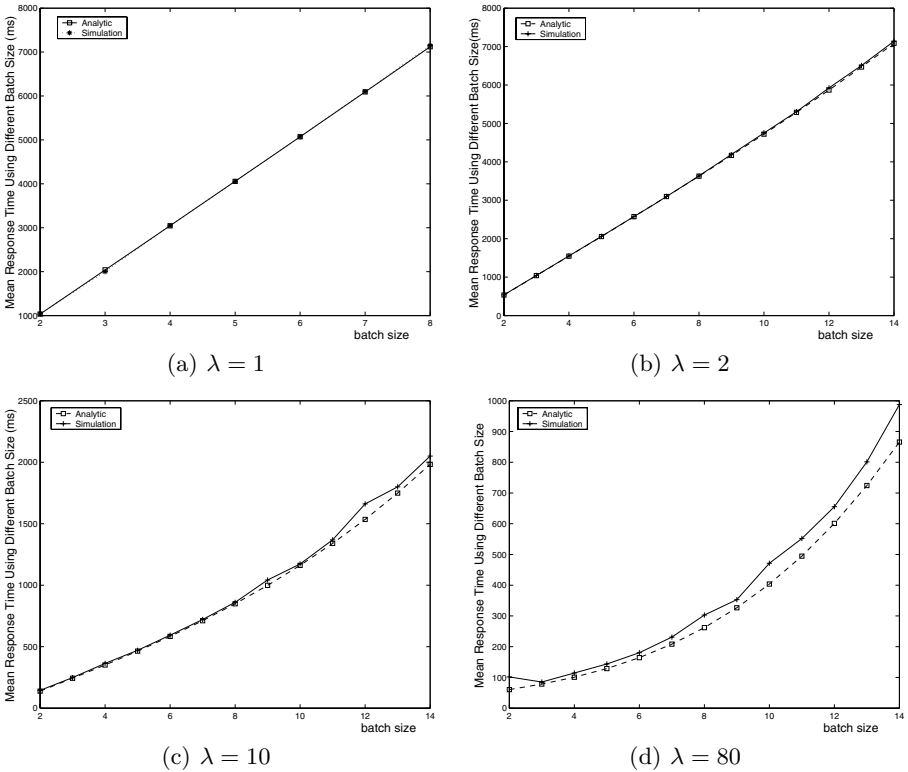


Fig. 3. Mean Response Time Validation with Optimal Batch Size

decryption request to decryption device. Otherwise, the mean response time in this case is not increased obviously. The solution of the optimal batch size are decreased with the  $\lambda$  when  $\lambda > 30$  approximately.

**Performance with Optimal Batch Size:** Figure 3 illustrates the analytic mean response time, our simulation mean response time and the the mean response time of SB scheme. As show in the figure 3, the mean response time  $T$  of a batched system behaves nicely when using the optimal solution of the batch size. When  $\lambda = 4$ ,  $T$  reaches Maximum and decreased with the  $\lambda$  when using optimal batch size. If we determine  $b = 4$  in the SB scheme [9], the mean response time of a batched system behaves poorly especially when the arrival rate does not exceed 1 requests/sec.

**Performance with Different Batch Size:** Figure 4(a)-(c) show that the mean response time is almost linearly with the batch size when the arrival rate is relatively small. This is due to the fact that the total mean responding time  $T = T_q + T_c + T_B$ , where  $T_c$  is the time for waiting other client in the same



**Fig. 4.** Mean Response Time Validation of the Analytic Model against Simulation Using Different Batch Size

batching. When the arrival rate is relatively small, the main contribution to  $T$  is made by  $T_c$ . It is evident that the time  $T_c$  is increased linearly with the batch size. Otherwise, the batching service time of the server  $T_B$  is also increased with the batch size. Therefore the mean response time is also increased with the batch size when the arrival rate is relatively large(i. e. ,  $\lambda = 80$ ).

**Performance Speed Against Non-batching Scheme:** A non-batched system becomes unstable when the arrival rate  $\lambda > 1/T_{rsa} = 1/0.032 = 31.25$  due to the fact that a non-batched system becomes unstable when arrival rate higher than service rate [10]. But with batching, the mean response time of a batch system behaves nicely even at high loads.

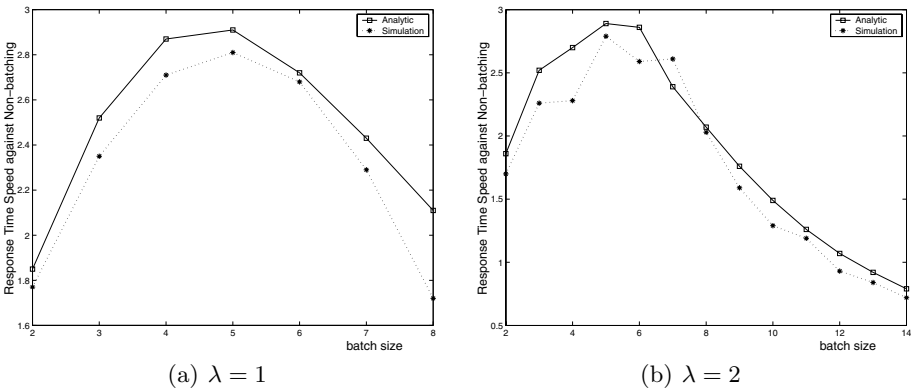
When the non-batching system is stable, the mean response time  $T'$  without batching in M/D/1 queue model while the mean service time  $\tau'$  is deterministic can be estimated as Eq 12 as [10] while the mean service time  $\tau'$  is deterministic. Since  $T_{rsa}$  is the majority of service time, the the mean service time  $\tau'$  of the server is roughly  $T_{rsa}$ .

$$T' = \tau' + \tau' \left( \frac{\tau' \lambda}{2(1 - \tau' \lambda)} \right) \tag{12}$$

Figure 5(a) and (b) illustrates the comparison of the mean response time of the batching schemes with the non-batched scheme. The vertical axis in each graph is the mean response time with different batch size divided by the mean response time with non-batched scheme.

From the analytic solution and simulation result of  $b$  described by the inequations 11 which is showed in figure 3, we can get  $b = 3$  when  $\lambda = 1$  and  $b = 6$  when  $\lambda = 2$ .

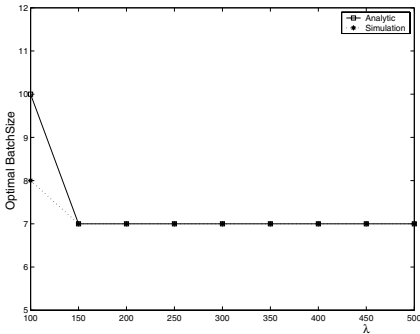
The speed of mean response time is an optimal one which equal to 2.52 approximately in figure 5(a). From figure 5(b), it is clear that speed of the mean response time is also optimal one which equal to 2.86 roughly. It is also clear that with the optimal batch size the batching system has considerable advantages whereas costs little.



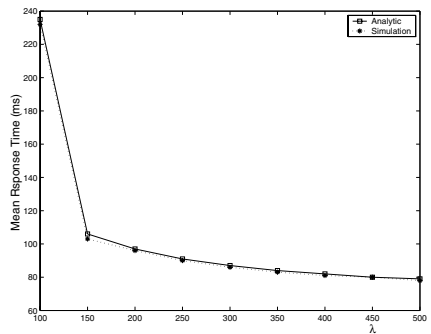
**Fig. 5.** Mean Response Time Speed Against Non-batching Scheme



**Multi-server Case Validation:** Based on the Theorem 1, a batched system becomes unstable when the arrival rate  $\lambda > 100$ . In other words, with analytical model and simulation, we can not find the result of solution of batch size when the arrival rate  $\lambda > 100$  requests/sec. In this case, we consider using two batch-servers to provide service for SSL handshake requests. In figure 6, we give the result of the optimal batch size using analytical and simulation. In figure 7, we give the result of mean response time using the optimal batch size in multi-server system.



**Fig. 6.** Optimal Batch Size in Multi-server System



**Fig. 7.** Mean Response Time Validation in Multi-server System

## 6 Conclusion

In this paper, we proposed the new method of assigning the set of public exponents  $e_i$  only using unique certificate in batching SSL handshake protocol. This solution improved the multiple certificates scheme without adding any modification on SSL protocol and additional information that needs to be sent to the clients .

We have shown the stability conditions for the batching SSL handshake and the optimal solution of batch size by developing an analytical model. We also validated this model with simulation results. We have also demonstrated that the mean response time etc. is very nice when using the optimal batch size.

## References

1. C. Coarfa, P. Druschel, and D. S. Wallach, "Performance Analysis of TLS Web Servers," NDSS 2002
2. T. Dierks, and E. Rescorla, "The TLS Protocol, Version 1.1," IETF Draft, RFC 2246, 2005
3. I. Goldberg, and D. Wagner, "Randomness and the Netscape Browser," *Dr. Dobb's Journal*, pp. 66-70, January 1996.

4. A. O. Freier, P. Karlton, P. C. Kocher, "The SSL Protocol, V3.0"
5. J. Feigenbaum, M. J. Freedman, T. Sander, A. Shostack, "Privacy Engineering for Digital Rights Management Systems," 2001 ACM Workshop on Security and Privacy in Digital Rights Management, LNCS 2320, pp.76-105, 2002.
6. E. Rescorla, A. Cain, and B. Korver, "SSLACC: A Clustered SSL Accelerator," Proceedings of the 11th USENIX Security Conference.
7. A. Goldberg, R. Buff, and A. Schmitt, "Secure Web Server Performance Dramatically Improved By Caching SSL Session Keys," in Workshop on Internet Server Performance, June 1998.
8. A. Fiat, "Batch RSA," Crypto'89, pp.175-185, 1989. See also Journal of Cryptology, 10(2):75-88, 1997
9. H. Shacham, and D. Boneh, "Improving SSL Handshake Performance via Batching," RSA'2001, Lecture Notes in Computer Science, Vol. 2020, pp.28-43, 2001.
10. L.Kleinrock. Queueing Systems, Volume I.Wiley-Interscience,1975.
11. W. C. Cheng, C. -F. Chou, and L. Golubchik, "Performance of Batch-based Digital Signatures," 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp.291-299, 2002.
12. R.Alec, Capitalism Is Dead: Peoplism Rules: Creating Success Out of Corporate Chaos, McGraw-Hill, 2003.
13. C.Vuillaume, Efficiency Comparison of Several RSA Variants,Studienarbeit, March 2003. <http://www.cdc.informatik.tu-darmstadt.de/reports/reports/studien.pdf>.
14. Cryptix:The Open Source Toolkit.<http://www.cryptix.org/>

# Achieving Efficient Conjunctive Keyword Searches over Encrypted Data

Lucas Ballard, Seny Kamara, and Fabian Monrose

Department of Computer Science,  
Johns Hopkins University,  
Baltimore, MD, USA  
{lucas, seny, fabian}@cs.jhu.edu

**Abstract.** We present two provably secure and efficient schemes for performing conjunctive keyword searches over symmetrically encrypted data. Our first scheme is based on Shamir Secret Sharing and provides the most efficient search technique in this context to date. Although the size of its trapdoors is linear in the number of documents being searched, we empirically show that this overhead remains reasonable in practice. Nonetheless, to address this limitation we provide an alternative based on bilinear pairings that yields constant size trapdoors. This latter construction is not only asymptotically more efficient than previous secure conjunctive keyword search schemes in the symmetric setting, but incurs significantly less storage overhead. Additionally, unlike most previous work, our constructions are proven secure in the standard model.

## 1 Introduction

Remote and untrusted storage systems [21,14,13] allow clients with limited resources to store and distribute large amounts of data at low cost. However, in order to preserve confidentiality, the remotely-stored data must be encrypted prior to transmission. Unfortunately, encryption restricts a client's ability to selectively access segments of her data, especially when she wishes to only retrieve specific content (e.g., related to a given keyword). To address this dilemma, a number of techniques have been recently proposed for achieving a less stringent storage model, one based on the notion of secure, delegated, searchable encryption (e.g., [28,17,12,10,29]). Intuitively, in order to provide secure searchable encryption schemes, most of these approaches associate an index with each document that, when combined with a trapdoor for a keyword, returns information signifying the association of the keyword with the document. Informally, such keyword searches are considered secure if they leak at most one bit of information about each document, namely, whether or not that document contains the keyword.

While the ability to perform single keyword searches is useful in some settings, clearly, it is more desirable to search on multiple keywords, and in particular, on boolean combinations of these keywords. Unfortunately, most previous

suggestions for permitting multiple keyword searches either fail to do so efficiently, or leak unnecessary information. To see why, consider that the suggested approaches for achieving this goal have been to either (i) send a trapdoor for each keyword and expect the server to return the set intersection (in the case of conjunctions) or the set union (in the case of disjunctions) of the matching documents (e.g., see [28,17]), or (ii) store information corresponding to every possible boolean combination of keywords on the server. The former suggestion leaks information that is linear in the number of conjuncts being searched, and the latter, while secure, incurs storage overhead that is exponential in the number of keywords associated with the document.

In this paper, we focus on providing provably-secure conjunctive keyword search over symmetrically encrypted data, while minimizing the computational and storage overhead imposed on both the client and server. To this end, we present two constructions, one based on a non-standard use of Shamir Secret Sharing [27] and another based on bilinear pairings. We show that our first construction is the most *practical* conjunctive keyword search scheme known to date. Although the size of the trapdoors it generates is linear in the number of documents being searched, our experiments show that this overhead remains manageable in practice. For situations where constant sized trapdoors are required, we provide an alternative construction that is the most *asymptotically* efficient scheme we are aware of (in terms of both space and time complexity) in the symmetric setting. Moreover, both of our constructions are provably secure in the standard model.

## 2 Related Work

Song, Wagner, and Perrig introduced the notion of searchable encryption in [28]. In that work, the authors present a new encryption algorithm that embeds extra information into the ciphertext such that when it is used in conjunction with a trapdoor, it discloses whether a particular keyword is stored in a document. Unfortunately, search requires computation linear in the size of each document and reveals statistical information about the distribution of the underlying plaintext.

Both of these shortcomings are addressed by the work of Goh [17], which presents a construction that uses per-document indexes derived from Bloom filters [7]. There, each word in the document is processed using a pseudo-random function and then inserted into a Bloom filter. The client then provides a trapdoor consisting of an indicator of which bits in the filter should be tested, thereby resulting in constant per-document search time. Moreover, Goh's work also introduced the notion of semantic security against chosen-keyword attacks (called IND-CKA), which is the first formal notion of security defined for searchable encryption.

As discussed earlier, neither of these schemes allow users to perform boolean keyword searches securely and efficiently. This shortcoming was first addressed by Golle, Staddon and Waters in [18], where they present two solutions that achieve the desired level of security. The first is provably secure under the De-

cision Diffie-Hellman assumption [8] and requires two modular exponentiations per document for searching. Additionally, the size of the trapdoors (referred to as search capabilities in [18]) is linear in the number of documents being searched. Although it is shown that portions of the trapdoors can be transmitted before a search even takes place, this overhead is still undesirable. Furthermore, it requires the client to know the number of searches she wishes to perform a priori. The second construction is based on bilinear pairings and is proven secure under a new hardness assumption. That scheme achieves constant sized trapdoors, but requires a linear number (with respect to keywords) of pairing computations per document in order to perform a search—an overhead that is arguably unrealistic, particularly in the presence of a large collection of documents.

More recently, Park, Kim and Lee proposed the first public-key searchable encryption schemes [10,29,15] that allow for secure conjunctive keyword searches [25]. Their constructions, based on the Bilinear Decision Diffie-Hellman (BDDH) and Bilinear Decision Diffie-Hellman Inversion (BDDHI) assumptions, have constant sized trapdoors and are more efficient than the schemes presented in [18]. Since the constructions presented in this paper are more efficient, in terms of computational and storage overhead, than both of the previous approaches, we argue that our schemes offer the most pragmatic choice in the symmetric setting.

### 3 Preliminaries

#### 3.1 Model

We assume the standard model for symmetric searchable encryption schemes (as used in [28,17,12,18]), which includes a client and a server that can be trusted to interact in a protocol, but not to abstain from attempting to learn information that is not explicitly released by the client. The client has a set of  $m$  documents  $\mathcal{D} = (D_1, \dots, D_m)$  that she wishes to store on the server in encrypted form, while still retaining the ability to search through them. To do so, she first generates an index  $\mathcal{I}_i$  for each document  $D_i$  and stores both the index and the encrypted document  $\mathcal{E}(D_i)$  on the server. Here, we assume that  $\mathcal{E}$  is an arbitrary symmetric encryption scheme, such as AES [16], and is independent of our index constructions. To search the document collection for a given keyword  $w$ , the client generates and sends a trapdoor  $\mathcal{T}$  to the server who proceeds to search each index for  $w$ . The server then returns the appropriate set of documents to the client.

In this work, we also assume the standard model for conjunctive keyword searches over encrypted data as presented in [18]. Namely, we work in the setting where each document is associated with a list of keywords. In particular, we make the following assumptions: (i) the number of keywords associated with a document remains fixed and (ii) no keyword appears at two different locations in a list. The first constraint can be satisfied by simply adding null keywords to the list, while the second can be satisfied by prepending each keyword with a field name or the value of a counter. As in [18,25], to reduce computational burden, trapdoors specify which positions should be searched within an index.

### 3.2 Notation

Throughout this paper we use the following notation. Let  $\Gamma$  be a dictionary of words, and  $2^\Gamma$  be the set of all possible documents. Further, let  $\mathcal{D} \subseteq 2^\Gamma$  be a collection of  $m$  documents  $\mathcal{D} = (D_1, \dots, D_m)$ . We associate a list of keywords  $W_i = (w_{i,1}, \dots, w_{i,n})$  with each document  $D_i$ . When the associated document is clear from the context, we simplify the notation as  $W_i = (w_1, \dots, w_n)$ .  $\mathcal{I}_i$  refers to an index for a document  $D_i$ , and  $\mathcal{T}_c$  to a trapdoor for a conjunction of  $d$  words  $c = (w_1, \dots, w_d)$ . If a trapdoor is composed of one term for each document in  $\mathcal{D}$  (i.e., is linear in the size of  $\mathcal{D}$ ), then we say that  $\mathcal{T}_c$  is composed of  $m$  tokens  $(T_1, \dots, T_m)$ .

Furthermore, we write  $x \stackrel{R}{\leftarrow} X$  to represent a random variable  $x$  being drawn uniformly from a set  $X$ . The output of a deterministic algorithm  $\mathcal{A}$  will be denoted by  $x \leftarrow \mathcal{A}$  and that of a probabilistic algorithm by  $x \stackrel{R}{\leftarrow} \mathcal{A}$ . Finally, let  $\text{negl}(k)$  denote a negligible function in  $k$ , and  $\parallel$  denote concatenation.

## 4 Definitions

In this section we reintroduce the definition of a secure conjunctive keyword search scheme and recall the notion of semantic security against chosen-keyword attacks.

**Definition 1 (Secure Conjunctive Keyword Search (SCKS)).** *Let  $\mathcal{W} = (W_1, \dots, W_m)$  be a collection of keyword lists. A SCKS scheme consists of four probabilistic polynomial-time algorithms:*

- **Keygen**( $1^k$ ): *is a probabilistic key generation algorithm that is executed by the client in order to instantiate the scheme. It takes as input a security parameter  $k$ , and returns a secret key  $K$ .*
- **BuildIndex**( $K, W_i$ ): *is executed by the client to construct an index. It takes as input a secret key  $K$  and a keyword list  $W_i$ . It returns  $W_i$ 's index  $\mathcal{I}_i$ .*
- **Trapdoor**( $K, \ell_1, \dots, \ell_d, w_1, \dots, w_d$ ): *is executed by the client to generate a trapdoor for a given conjunction of keywords. It takes as input a secret key  $K$ , the locations in the index to search  $(\ell_1, \dots, \ell_d)$ , and a list of  $d$  conjuncts  $(w_1, \dots, w_d)$ . It returns a trapdoor  $\mathcal{T}_c$  for the conjunction  $c = (w_1 \wedge \dots \wedge w_d)$ .*
- **SearchIndex**( $\mathcal{I}_i, \mathcal{T}_c$ ): *is executed by the server on behalf of the client to search for the occurrence of a conjunction in an index. It takes as input an index  $\mathcal{I}_i = \text{BuildIndex}(K, W_i)$ , where  $W_i = (w_1, \dots, w_n)$ , and a trapdoor  $\mathcal{T}_c$  for a conjunction  $c = (w'_1 \wedge \dots \wedge w'_d)$ . It returns **true** if  $w'_j = w_{\ell_j}$  for  $1 \leq j \leq d$ ; and **false** otherwise.*

Intuitively, the notion of security we seek to capture can be summarized as follows: given access to a set of indexes, a server should not be able to learn any partial information about the associated keyword lists that he cannot learn from a trapdoor that was *explicitly* given to him by the client. Note that in the context of conjunctive keyword searches, this implies that the trapdoor for a

given conjunction  $c = (w_1 \wedge \dots \wedge w_d)$  should not help the server in generating a trapdoor for any other conjunction  $c' = (w'_1 \wedge \dots \wedge w'_d)$ , even if  $\{w'_1, \dots, w'_d\} \subset \{w_1, \dots, w_d\}$ . In addition, this notion of security should hold even against a server that can mount chosen keyword attacks, or in other words, against a server that can trick the client into generating trapdoors for keywords of its choice. More formally, this notion of security is known as semantic security against chosen keyword-attacks as introduced in [17]. Golle, Staddon and Waters [18] present three games that formally capture semantic security against chosen-keyword attacks for SCKS schemes and show that they are all asymptotically equivalent. In this work, we only make use of two of these games, namely *indistinguishability of ciphertext from ciphertext* and *indistinguishability of ciphertext from random*, which we briefly review below.

**Definition 2 (Indistinguishability of ciphertext from ciphertext (ICC) [18]).** For all probabilistic polynomial-time adversaries  $\mathcal{A}$ :

$$\Pr \left[ \begin{array}{l} b' = b : K \leftarrow \text{Keygen}(1^k); \\ (W_0, W_1) \leftarrow \mathcal{A}^{\text{BuildIndex}(K, \cdot), \text{Trapdoor}(K, \cdot)}; \\ b \stackrel{R}{\leftarrow} \{0, 1\}; \mathcal{I}_b \leftarrow \text{BuildIndex}(K, W_b); \\ b' \leftarrow \mathcal{A}^{\text{BuildIndex}(K, \cdot), \text{Trapdoor}(K, \cdot)}(\mathcal{I}_b) \end{array} \right] \leq \frac{1}{2} + \text{negl}(k)$$

with the restriction that  $\mathcal{A}$  must choose  $(W_0, W_1)$  such that  $|W_0| = |W_1|$  and such that  $\text{SearchIndex}(\mathcal{I}_0, \mathcal{T}) = \text{SearchIndex}(\mathcal{I}_1, \mathcal{T})$  for all  $\mathcal{T}$  generated using its Trapdoor oracle.

**Definition 3 (Indistinguishability of ciphertext from random (ICR) [18]).** For all probabilistic polynomial-time adversaries  $\mathcal{A}$ :

$$\Pr \left[ \begin{array}{l} b' = b : K \leftarrow \text{Keygen}(1^k); \\ W_0 \leftarrow \mathcal{A}^{\text{BuildIndex}(K, \cdot), \text{Trapdoor}(K, \cdot)}; W_1 \stackrel{R}{\leftarrow} 2^{\Gamma}; \\ b \stackrel{R}{\leftarrow} \{0, 1\}; \mathcal{I}_b \leftarrow \text{BuildIndex}(K, W_b); \\ b' \leftarrow \mathcal{A}^{\text{BuildIndex}(K, \cdot), \text{Trapdoor}(K, \cdot)}(\mathcal{I}_b) \end{array} \right] \leq \frac{1}{2} + \text{negl}(k)$$

with the restriction that  $W_1$  must be chosen such that  $|W_0| = |W_1|$  and such that  $\text{SearchIndex}(\mathcal{I}_0, \mathcal{T}) = \text{SearchIndex}(\mathcal{I}_1, \mathcal{T})$  for all  $\mathcal{T}$  generated using its Trapdoor oracle.

**Theorem 1 ([18]).** If there exists an adversary  $\mathcal{A}$  that wins game ICC with non-negligible probability, then there exists another adversary  $\mathcal{B}$  that wins game ICR with the same probability.

## 5 A Construction Based on Secret Sharing

In this section we describe our first secure conjunctive keyword search scheme, denoted as SCKS-SS. It is based on Shamir’s threshold secret sharing scheme

(SSS) and is provably secure in the standard model. We note that this construction does not take advantage of SSS's threshold properties, which suggests that similar constructions could be achieved using random oracles or other secret sharing schemes. We chose SSS due to its universal familiarity and efficiency.

In [27], Shamir proposed a  $(k, n)$ -threshold secret sharing scheme based on polynomial interpolation in the field  $\mathbb{Z}_p$ . Informally, the scheme is composed of two algorithms  $\text{SSS} = (\text{share}, \text{recover})$  defined as follows. If  $S \in \mathbb{Z}_p$  is a secret value we wish to share between  $n$  players, and if we require that at least  $k$  shares are needed to recover  $S$ ,  $\text{share}$  operates as follows: a dealer generates a random  $k - 1$  degree polynomial  $\mathcal{P}$  such that  $\mathcal{P}(0) = S$ ; it then chooses  $n$  random points on  $\mathcal{P}$  and distributes them as shares. In order to recover the secret  $\mathcal{P}(0) = S$ , the  $\text{recover}$  algorithm simply requires that at least  $k$  players pool their shares, perform standard polynomial interpolation, and evaluate the resulting polynomial at 0. The unconditional security of SSS follows from the fact that one cannot interpolate a  $k - 1$  degree polynomial with fewer than  $k$  points.

## 5.1 Our Construction

Our construction makes use of a pseudo-random function  $f : \{0, 1\}^k \times \{0, 1\}^l \rightarrow \mathbb{Z}_p \times \mathbb{Z}_p$ , where  $l$  is the length of the longest word in  $\Gamma$ . Each keyword list has an associated identifier,  $i$ , which is *never* reused. In particular, even if two lists have the same set of keywords at the same locations, their identifier will be different. Given a list  $W_i$ , we write  $f_K^i(w)$ , where  $w \in W_i$ , to refer to the following operation:  $f(K, i || w)$ . Let  $\mathcal{W} = (W_1, \dots, W_m)$  be a collection of  $m$  keyword lists, each composed of  $n$  words.  $\text{SCKS-SS} = (\text{Keygen}, \text{BuildIndex}, \text{Trapdoor}, \text{SearchIndex})$  is then given as follows:

- $\text{Keygen}(1^k)$ : Generate a secret key  $K \xleftarrow{R} \{0, 1\}^k$  and a random prime  $p > n$ .
- $\text{BuildIndex}(K, W_i)$ : For each word  $w_j \in W_i$ , let  $\sigma_j = f_K^i(w_j) = (x_j, y_j)$ . Output  $\mathcal{I}_i = (\sigma_1, \dots, \sigma_n)$ .
- $\text{Trapdoor}(K, \ell_1, \dots, \ell_d, w'_1, \dots, w'_d)$ : For each  $W_i \in \mathcal{W}$ , let  $S_i = \text{recover}(f_K^i(w'_1), \dots, f_K^i(w'_d))$ . Output  $\mathcal{T} = (S_1, \dots, S_m, \ell_1, \dots, \ell_d)$ .
- $\text{SearchIndex}(\mathcal{I}_i, \mathcal{T})$ : If  $S_i = \text{recover}(\sigma_{\ell_1}, \dots, \sigma_{\ell_d})$ , then output true, otherwise output false.

## 5.2 Security

Since correctness follows from the description of the scheme, we provide only a proof of security. To show that SCKS-SS is semantically secure against chosen-keyword attacks, we first state two useful lemmas. Due to space considerations, we omit the proofs of these lemmas, but refer the reader to the full version of this paper [3]. The first states that given two arbitrary keyword lists, their indexes are independent to any probabilistic polynomial-time adversary. We note that this holds even if the lists are composed of the same keywords. The second lemma states that given two different conjunctions, their corresponding trapdoors will



be independent (also to any probabilistic polynomial-time adversary), even if they are generated in order to search the same keyword list.

The usefulness of these lemmas will become apparent in our main theorem (Theorem 2), where we claim that since the adversary cannot learn anything about its challenge index from access to its **BuildIndex** and **Trapdoor** oracles, we need only consider semantic security against chosen plaintext attacks.

**Lemma 1.** *Let  $W_a$  and  $W_b$  be two keyword lists such that  $a \neq b$ . If  $f_K$  is a pseudo-random function, then  $\mathcal{I}_a$  is independent of  $\mathcal{I}_b$  for all probabilistic polynomial-time adversaries.*

**Lemma 2.** *Let  $\mathcal{W} = (W_1, \dots, W_m)$  be a collection of  $m$  keyword lists and let  $c = (w_1 \wedge \dots \wedge w_d)$  and  $c' = (w'_1 \wedge \dots \wedge w'_\delta)$  be two conjunctions of length  $d$  and  $\delta$ , respectively. If  $f_K$  is a pseudo-random function and if  $c \neq c'$ , then for any probabilistic polynomial-time adversary  $\mathcal{A}$ ,  $\mathcal{I}_c$  is independent of  $\mathcal{I}_{c'}$ .*

**Theorem 2.** *If  $f_K$  is a pseudo-random function, then SCKS-SS is semantically secure against chosen-keyword attacks.*

*Proof.* Since the identifier of a keyword list is never repeated, we know from Lemma 1 that if  $f_K$  is a pseudo-random function then no two indexes will be correlated even if they contain the same keywords. It follows that  $\mathcal{A}$  cannot learn anything about its ICC challenge index  $\mathcal{I}_b$  from any of the indexes returned by its **BuildIndex** oracle.

Furthermore, consider the restriction imposed on  $\mathcal{A}$ 's choice of keyword lists in the ICC game, namely that it must choose two lists  $W_0$  and  $W_1$ , such that  $\text{SearchIndex}(\mathcal{I}_0, \mathcal{T}) = \text{SearchIndex}(\mathcal{I}_1, \mathcal{T})$  for all trapdoors  $\mathcal{T}$  returned by its **Trapdoor** oracle. This implies that any such trapdoor will be useless to  $\mathcal{A}$  in distinguishing whether  $\mathcal{I}_b$  is the index for  $W_0$  or  $W_1$ . In addition, by Lemma 2, we know that if  $f_K$  is a pseudo-random function, then trapdoors generated for different conjunctions are independent. Taken together, the two preceding statements imply that  $\mathcal{A}$  cannot use the results of its **Trapdoor** queries to either search over its challenge index  $\mathcal{I}_b$ , or to generate any new trapdoors with which it can try to search over  $\mathcal{I}_b$ .

From the previous discussion, it is then safe to only consider  $\mathcal{A}$ 's challenge index. If  $f$  is a random function, then  $\mathcal{I}_0$  and  $\mathcal{I}_1$  are independent of  $W_0$  and  $W_1$ , respectively. If we replace  $f$  by a pseudo-random function  $f_K$ , then to any probabilistic polynomial-time adversary,  $\mathcal{I}_b$  will be independent of its associated keyword list  $W_b$ . Thus  $\mathcal{A}$  will not be able to distinguish between  $W_0$  and  $W_1$  given  $\mathcal{I}_b$ , otherwise we could build an adversary  $\mathcal{B}$  that could distinguish between  $f_K$  and a random function.  $\square$

## 6 A Construction Based on Bilinear Maps

Our second construction, SCKS-XDH, achieves constant transmission overhead at the cost of placing a larger computational burden on the server. The security

of the scheme is based on a new variant of the External Diffie-Hellman (XDH) assumption. The XDH assumption was first introduced in [26] and formalized in [9,2]. It has been used more recently as a hardness assumption in [?,11]

**Assumption 1 (Decision Diffie-Hellman (DDH)).** *Let  $\langle g \rangle = \mathbb{G}$  be a cyclic group of order  $p$  and  $c \stackrel{R}{\leftarrow} \mathbb{Z}_p$ . The Decision Diffie-Hellman assumption holds in  $\mathbb{G}$  if no probabilistic polynomial-time adversary  $\mathcal{A}$  can distinguish between tuples  $(g, g^a, g^b, g^{ab})$  and  $(g, g^a, g^b, g^c)$ , with probability non-negligibly greater than  $\frac{1}{2}$ :*

$$|\Pr [\mathcal{A}(g^a, g^b, g^{ab}) = 1] - \Pr [\mathcal{A}(g^a, g^b, g^c) = 1]| \leq \frac{1}{2} + \text{negl}(|p|)$$

**Assumption 2 (External Diffie-Hellman (XDH) [2]).** *Let  $\langle P \rangle = \mathbb{G}_1$  and  $\langle Q \rangle = \mathbb{G}_2$  be two disjoint cyclic subgroups of order  $p$  of an elliptic curve, and let  $\hat{e}$  be a non-degenerate bilinear map  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ . The XDH assumption holds across  $\mathbb{G}_1$  and  $\mathbb{G}_2$  if the DDH assumption holds within  $\mathbb{G}_1$ .*

A concrete algebraic setting where this assumption holds is discussed further in [2]. To prove the security of our construction, we make use of a slightly stronger variant of the XDH assumption which we call the *Mixed XDH assumption*.

**Assumption 3 (Mixed External Diffie-Hellman (MXDH)).** *Let  $\langle P \rangle = \mathbb{G}_1$  and  $\langle Q \rangle = \mathbb{G}_2$  be two groups of order  $p$ , and let  $\hat{e}$  be a non-degenerate bilinear map  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ . The MXDH assumption holds across  $\mathbb{G}_1$  and  $\mathbb{G}_2$  if given  $(P, aP, bP, cP, aQ, bQ)$ , no probabilistic polynomial-time adversary  $\mathcal{A}$  can distinguish between tuples  $(P, aP, bP, abP)$  and  $(P, aP, bP, cP)$ , where  $c \stackrel{R}{\leftarrow} \mathbb{Z}_p$  (i.e., the DDH assumption holds within  $\mathbb{G}_1$ ).*

### 6.1 Our Construction

Let  $\langle P \rangle = \mathbb{G}_1$  and  $\langle Q \rangle = \mathbb{G}_2$  be two groups of prime order  $p$  and  $\hat{e}$  be a non-degenerate bilinear map  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  such that the XDH assumption holds across  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . Additionally, let  $f : \{0, 1\}^k \times \{0, 1\}^l \rightarrow \mathbb{Z}_p$  be a pseudo-random function, where  $l$  is the length of the longest word in  $\Gamma$ . Recall that in our model we assume that all keywords are distinct and that this can be achieved by simply concatenating the value of a counter to each keyword. We define SCKS-XDH = (Keygen, BuildIndex, Trapdoor, SearchIndex) as follows:

- **Keygen**( $1^k$ ): Generate a secret key  $K \stackrel{R}{\leftarrow} \{0, 1\}^k$ , and choose two points  $P$  and  $Q$  such that  $\langle P \rangle = \mathbb{G}_1$  and  $\langle Q \rangle = \mathbb{G}_2$ .  $Q$  is kept private.
- **BuildIndex**( $K, W_i$ ): Choose  $r_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ . For each word  $w_j \in W_i$ , let  $s_j = f_K(w_j)$ . Output  $\mathcal{I}_i = (r_i P, r_i s_1 P, \dots, r_i s_n P)$ .
- **Trapdoor**( $K, \ell_1, \dots, \ell_d, w'_1, \dots, w'_d$ ): Choose  $\rho \stackrel{R}{\leftarrow} \mathbb{Z}_p$ . Let  $t = \left(\rho \sum_{j=1}^d f_K(w'_j)\right) Q$ . Output  $\mathcal{T} = (t, \rho Q, \ell_1, \dots, \ell_d)$ .
- **SearchIndex**( $\mathcal{I}_i, \mathcal{T}$ ): If  $\hat{e}(t, r_i P) = \hat{e}(\rho Q, \sum_{j=1}^d r_i s_{\ell_j} P)$ , then output true, otherwise output false.

To demonstrate correctness, consider a user that wishes to search for a conjunction  $c = (w'_1 \wedge \dots \wedge w'_d)$  over the index locations specified by the sequence of integers  $(\ell_1, \dots, \ell_d)$ . Let  $\mathcal{T} = \text{Trapdoor}(K, \ell_1, \dots, \ell_d, w'_1, \dots, w'_d)$  be the trapdoor enabling the server to search for  $c$ . Furthermore, let  $W_i = (w_1, \dots, w_n)$  be a keyword list, and  $\mathcal{I}_i = \text{BuildIndex}(K, W_i)$  be its index. If  $W_i$  is such that  $w_{\ell_j} = w'_j$  for  $1 \leq j \leq d$  (i.e.  $W_i$  includes all the words in the conjunction  $c$  at the appropriate locations) then:  $\hat{e}(t, r_i P) = \hat{e}((\rho \sum_{j=1}^d f_K(w'_j))Q, r_i P) = \hat{e}(Q, P)^{r_i \rho \sum_{j=1}^d f_K(w'_j)} = \hat{e}(Q, P)^{\rho \sum_{j=1}^d r_i s_{\ell_j}} = \hat{e}(\rho Q, \sum_{j=1}^d r_i s_{\ell_j} P)$  and  $\text{SearchIndex}(\mathcal{I}_i, \mathcal{T})$  will return true.

### 6.2 Security

**Theorem 3.** *If the Mixed XDH assumption holds, then SCKS-XDH is semantically secure against chosen-keyword attacks.*

*Proof.* Given an adversary  $\mathcal{A}$  that wins the ICR game with non-negligible probability over  $\frac{1}{2}$ , we describe an adversary  $\mathcal{B}$  that uses  $\mathcal{A}$  to break the MXDH assumption with the same probability. By theorem 1 this implies that there exists another adversary that can win game ICC also with non-negligible probability over  $\frac{1}{2}$ . Let  $\langle P \rangle = \mathbb{G}_1$ ,  $\langle Q \rangle = \mathbb{G}_2$ , and  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  be an appropriate XDH setting, and let  $(P, aP, bP, cP, aQ, bQ)$  be  $\mathcal{B}$ 's MXDH challenge.  $\mathcal{B}$ 's goal is to solve the DDH problem in  $\mathbb{G}_1$ , or in other words to decide whether  $c = ab$ .

$\mathcal{B}$  begins by simulating  $\mathcal{A}$ . To start,  $\mathcal{A}$  will make a polynomial number of  $\text{BuildIndex}$  queries which  $\mathcal{B}$  will answer as follows. Let  $\mathcal{A}$ 's query keyword list be  $W_i = (w_{i,1} \wedge \dots \wedge w_{i,n})$ , where  $0 \leq i \leq \text{poly}(k)$ .  $\mathcal{B}$  chooses a value  $r_i \xleftarrow{R} \mathbb{Z}_p$  and for each word  $w_{i,j} \in W_i$ , where  $1 \leq j \leq n$ , it picks a random value  $z_{i,j} \xleftarrow{R} \mathbb{Z}_p$  and computes  $r_i z_{i,j} bP$ . To be consistent across different queries,  $\mathcal{B}$  keeps track of the correspondence between keywords  $w_{i,j} \in W_i$  and the random  $z_{i,j}$  values it chooses. Notice that if  $\mathcal{B}$  is given  $bP$  as part of its MXDH challenge, it can compute  $r_i z_{i,j} bP$ . Finally, it returns the index  $\mathcal{I}_i = (r_i P, r_i z_{i,1} bP, \dots, r_i z_{i,n} bP)$  to  $\mathcal{A}$ .

When  $\mathcal{A}$  makes a  $\text{Trapdoor}$  query with index locations  $(\ell_1, \dots, \ell_d)$  and conjunction  $c = (w'_1 \wedge \dots \wedge w'_d)$ ,  $\mathcal{B}$  begins by choosing a random value  $\rho \xleftarrow{R} \mathbb{Z}_p$ . It then computes  $t = \rho(\sum_{\lambda=1}^d \gamma_\lambda bQ)$  where  $\gamma_\lambda = z_{i,j}$  if  $w'_\lambda$  previously appeared at some position in one of  $\mathcal{A}$ 's queries, and  $\gamma_\lambda \xleftarrow{R} \mathbb{Z}_p$  otherwise. Observe that while  $\mathcal{B}$  does not know  $b$ , it can compute  $z_{i,j} bQ$  (and thus  $t$ ), since  $bQ$  is part of its MXDH challenge.

Finally,  $\mathcal{B}$  returns the trapdoor  $\mathcal{T} = (t, \rho Q, \ell_1, \dots, \ell_d)$  to  $\mathcal{A}$ . Note that  $\mathcal{T}$  is a valid trapdoor for  $c = (w'_1 \wedge \dots \wedge w'_d)$ , and in particular, that since  $\mathcal{B}$  consistently uses the same value  $z_{i,j}$  for word  $w_j$ ,  $\text{SearchIndex}(\mathcal{I}_i, \mathcal{T})$  will return true if and only if  $W_i$  includes all the words in the conjunction  $c$  at the locations specified by  $(\ell_1, \dots, \ell_d)$ .

After polynomially many  $\text{BuildIndex}$  and  $\text{Trapdoor}$  queries,  $\mathcal{A}$  submits a keyword list  $W^* = (w_1^*, \dots, w_n^*)$ .  $\mathcal{B}$  returns to  $\mathcal{A}$  the challenge index  $\mathcal{I}^* = (aP, \gamma_1^* cP, \dots, \gamma_n^* cP)$ , where  $\gamma_\lambda^* = z_{i,j}$  if  $w_\lambda^*$  previously appeared in one of  $\mathcal{A}$ 's queries to

either its `BuildIndex` or `Trapdoor` oracles, and  $\gamma_\lambda^* \xleftarrow{R} \mathbb{Z}_p$  otherwise. Observe that if  $c = ab$ , then  $\mathcal{I}^*$  is a correct index for  $W^*$ , while if  $c \neq ab$  then  $\mathcal{I}^*$  is a correct index for some other arbitrary keyword list. In particular, if  $c$  is random then  $\mathcal{I}^*$  is an index for a random set of keywords. After the challenge,  $\mathcal{A}$  is allowed to make more `Trapdoor` and `BuildIndex` queries (with the same restrictions as before), which  $\mathcal{B}$  answers as it did in the previous steps.

Finally,  $\mathcal{A}$  outputs a bit  $\beta$  that represents its decision as to whether  $\mathcal{I}^*$  is an index for  $W^*$  or some random keyword list.  $\mathcal{B}$  then returns  $\beta$  as its own answer to its MXDH challenge. It follows that  $\mathcal{B}$ 's probability in breaking its MXDH challenge is equal to  $\mathcal{A}$ 's probability in breaking the ICR game, which we assumed holds with non-negligible probability over  $\frac{1}{2}$ .  $\square$

## 7 Efficiency

COMPARISON WITH PREVIOUS CONSTRUCTIONS. We now consider the computational and space complexity of our constructions. In particular, we compare their efficiency to the work of Park, Kim and Lee [25]. We note that while the schemes in [25] are in the public-key setting, they are still more efficient than previous symmetric constructions presented in [18]. Since any public-key searchable encryption scheme can be converted to a symmetric one, we focus our comparison with the schemes in [25].

We refer to the first and second constructions in [25] as PKL-1 and PKL-2, respectively. Recall that  $m$  denotes the number of documents stored on the server,  $n$  the number of keywords associated with each document, and  $d$  the number of keywords comprising a search. Unless otherwise specified, measurements are in terms of the requirements to process *all* documents on the server.

In what follows we discuss the running time and storage overhead for all relevant operations, including building indexes, generating trapdoors, searching and storing indexes, and sending trapdoors. First, we note that SCKS-SS is the most computationally efficient construction for index generation and searching, requiring only  $m$  polynomial interpolations on  $d$  points. Furthermore, though both the size of its trapdoors and the running time of the trapdoor generation algorithm are linear in the number of documents, we show that this overhead still remains practical. Storage requirements for indexes are less than previous approaches [25,18], requiring only  $2mn$  points in  $\mathbb{Z}_p$  (where  $p$  is only 128 bits). We also note that SCKS-XDH incurs significantly less storage and transmission overhead than PKL-1 and PKL-2 as it need not store or send elements in integer groups. Additionally, SCKS-XDH is more efficient than both PKL-1 and PKL-2 for `BuildIndex` as it requires only  $m(n+1)$  multiplications in  $\mathbb{G}_1$ , is faster than PKL-1 for trapdoor generation, and is comparable to PKL-2 (but slower than PKL-1) for `SearchIndex`.

EMPIRICAL EVALUATION OF SCKS-SS AND SCKS-XDH. To evaluate the feasibility of our constructions, we implemented and benchmarked the relevant operations. All tests were performed on a 3.0 GHz Pentium 4 machine running Fedora

Core 3 Linux. Our implementations are in C++ and made use of the MIRACL [24] library for multi-precision and curve-based operations, and OpenSSL [23] for all other cryptographic primitives.

Our implementation of SCKS-SS performs all operations in the field  $\mathbb{Z}_p$ , where  $p$  is a 128-bit prime. We chose to instantiate  $f$  using HMAC-SHA1 [6]. In particular, we let  $f_K^i(w) = (h_K(w||i||0), h_K(w||i||1))$  where  $h_K(\cdot)$  denotes the last 128 bits of HMAC-SHA1's output under the key  $K$ . And though polynomial interpolation can be done as efficiently as  $O(d \log^2 d)$  [20], we use standard Lagrangian interpolation [20], which is  $O(d^2)$  to generate trapdoors and perform searches. Our implementation of SCKS-XDH uses a 160 bit curve, which gives security comparable to that of Diffie-Hellman in 1024 bit integer groups [22]. Due to the algebraic setting of the MXDH assumption, we can represent points in  $\mathbb{G}_1$  using 161 bits, but require 481 bits for points in  $\mathbb{G}_2$  [5]. Accordingly, operations in  $\mathbb{G}_2$  are slower than in  $\mathbb{G}_1$ <sup>1</sup>.

To evaluate the efficacy of our constructions, we present the time required to create 10,000 indexes of 10 keywords each, to generate trapdoors, and to search these indexes. In both constructions index generation grows linearly in the number of keywords. SCKS-SS requires slightly less than 2 seconds to generate 10,000 indexes with 10 keywords each, while SCKS-XDH requires 445 seconds.

The SCKS-SS operations that require interpolation, namely *Trapdoor* and *SearchIndex*, incur time that is quadratic in the number of keywords being searched. We note, however, that according to [19], user queries on the Web typically contain at most 3 keywords. If we assume such a setting, SCKS-SS is able to search 10,000 files in about 0.86 seconds, while trapdoor generation requires less than 1.5 seconds. The time required for SCKS-XDH to generate trapdoors and search a given index is essentially constant. Trapdoor generation requires 111 ms while searching 10,000 indexes requires approximately 720 seconds.

Although SCKS-SS requires trapdoors linear in the number of documents, since each token is only 16 bytes long, a trapdoor for 10,000 documents is merely 156 KB in size. The space required to store the indexes associated with a collection of 10,000 documents of 10 keywords each is 3.1 MB—which is much more space efficient than any previously known construction.

Although SCKS-XDH is less efficient in terms of index generation and searching than SCKS-SS, it requires less storage and only incurs constant transmission overhead. In fact, to store an index, the server need only keep a point in  $\mathbb{G}_1$  for each keyword of each document. As such, the indexes associated to a collection of 10,000 documents with 10 keywords each can be stored in approximately 2.1 MB. Also, since trapdoors are pairs of points in  $\mathbb{G}_2$  they can be represented in 0.12 KB.

*Acknowledgements.* The authors thank Avi Rubin for fruitful discussions and Eu-Jin Goh for comments on an earlier draft of this paper. This work was supported in part by a Bell Labs Graduate Research Fellowship and NSF award 0430338.

<sup>1</sup> Although SCKS-XDH could benefit from reusing line function coefficients in Miller's algorithm [4] (as suggested in [25]) we did not implement this optimization.

## References

1. G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable rfid tags via insubvertible encryption. In *12th ACM Conference on Computer and Communications Security (CCS 2005)*, 2005.
2. L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation resistant storage. Technical Report TR-SP-BGMM-050507, Johns Hopkins University Department of Computer Science, 2005.
3. L. Ballard, S. Kamara, and F. Monrose. Achieving efficient conjunctive keyword searches over encrypted data. Technical Report TR-SP-BKM-050920, Johns Hopkins University Department of Computer Science, 2005.
4. P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–369. Springer-Verlag, 2002.
5. P. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In *Selected Areas in Cryptography - SAC '03*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer-Verlag, 2004.
6. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.
7. B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
8. D. Boneh. The Decision-Diffie Hellman Problem. In *Third International Symposium on Algorithmic Number Theory (ANTS-III)*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998.
9. D. Boneh, X. Boyen, and H. Saham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.
10. D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology - Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer-Verlag, 2004.
11. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *Advances in Cryptology - EUROCRYPT 2005*, *Lecture Notes in Computer Science*, pages 302–321. Springer, 2005.
12. Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Third International Conference on Applied Cryptography and Network Security (ACNS 2005)*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455. Springer-Verlag, 2005.
13. I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: a distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66, 2001.
14. Microsoft Corp. Federated, available and reliable storage for an incompletely trusted environment (Farsite). See <http://research.microsoft.com/sn/Farsite/>.
15. D. Davis, F. Monrose, and M. Reiter. Time-scoped searching of encrypted audit logs. In *6th International Conference on Information and Communications Security (ICICS 2004)*, volume 3269 of *Lecture Notes in Computer Science*, pages 532–545. Springer-Verlag, 2004.
16. Federal Information Processing Standards. Advanced Encryption Standard (AES) – FIPS 197, November 2001.

17. E-J. Goh. Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003. See <http://eprint.iacr.org/2003/216>.
18. P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Applied Cryptography and Network Security Conference (ACNS)*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2004.
19. B. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.
20. D. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, Reading, Mass., 2nd edition, 1981.
21. J. Kubiawicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gum-madi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. In *Ninth international conference on Architectural support for programming languages and operating systems*, pages 190–201. ACM Press, 2000.
22. A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In *3rd International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465. Springer-Verlag, 2000.
23. The OpenSSL Library. See <http://www.openssl.org>.
24. Shamus Software Ltd. Multiprecision integer and rational arithmetic C/C++ library (MIRACL). See <http://indigo.ie/~mscott>.
25. D. Park, K. Kim, and P. Lee. Public key encryption with conjunctive field key-word search. In *5th International Workshop on Information Security Applications (WISA 2004)*, volume 3325 of *Lecture Notes in Computer Science*, pages 73–86. Springer-Verlag, 2004.
26. M. Scott. Authenticated ID-based key exchange and remote log-in with simple to ken and PIN number. Technical Report 2002/164, International Association for Cryptological Research, 2002. Available at <http://eprint.iacr.org/2002/164>.
27. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.
28. D. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In *IEEE Symposium on Research in Security and Privacy*, pages 44–55. IEEE Computer Society, May 2000.
29. B. Waters, D. Balfanz, G. Durfee, and D. Smetters. Building an encrypted and searchable audit log. In *Network and Distributed System Security Symposium (NDSS 2004)*. The Internet Society, 2004.

# Total Disclosure of the Embedding and Detection Algorithms for a Secure Digital Watermarking Scheme for Audio

David Megías, Jordi Herrera-Joancomartí, and Julià Minguillón

Estudis d'Informàtica i Multimèdia,  
Universitat Oberta de Catalunya,  
Av. Tibidabo 39–43, 08035 Barcelona  
Tel. (+34) 93 253 7523, Fax (+34) 93 417 6495  
{dmegias, jordiherrera, jminguillona}@uoc.edu

**Abstract.** This paper discusses the modification of a robust digital audio watermarking scheme to allow the disclosure of the embedding and detection algorithms. The chosen scheme uses MPEG 1 Layer 3 compression to determine the position of the mark bits in the frequency domain. The marking positions would be exposed if the original embedding algorithm was disclosed. In fact, it is shown that even if an attacker did not know the exact tuning parameters used for embedding, he or she could still produce an approximate superset of the marking frequencies from only a marked copy and successfully attack the file. To avoid this problem, a secret key is introduced in the embedding and detection processes. The secret key includes the seed of a pseudo-random number generator which is used to compute the exact marking positions. The modification is then analysed in terms of capacity, imperceptibility, robustness and security. The experiments show that the modified scheme preserves most of the properties of the original one, such as robustness against MP3 compression for the most frequently used bit rates, and does introduce additional security as the mark is more difficult to erase when the embedding and detection algorithms are disclosed.

**Keywords:** Audio Watermarking, Information Hiding, Intellectual Property Protection.

## 1 Introduction

Digital watermarking deals with the problem of embedding information (a mark) into a digital object (the cover object). Depending on the application, digital watermarking has different goals. For instance, for copyright protection, the embedded mark should not be removed when modifying the cover object unless the cover object itself becomes unusable. For authentication purposes, the watermark should be fragile in the sense that a minor change in the cover object should produce the loss of the mark.

The first few digital watermarking applications were focused on the copyright protection problem. Within this scenario, the major concern about watermarking was robustness, since the embedded mark should not be able to be removed. To measure the robustness of watermarking schemes, different benchmark tools were developed



(for example [6]) and the schemes were exhaustively tested against the attacks included in those benchmarks. At this time, no existing watermarking scheme supports the vast range of attacks included in all those benchmarks. However, a deep study on the survived attacks shows that some of the suggested watermarking schemes are robust against a moderate number of attacks. In fact, the survived attacks are enough for many specific applications. In the light of these results, the next benchmark generation is focused on defining application-oriented benchmarks [15].

At this first stage, the problem of removing the mark from a marked object was only dealt from the robustness point of view. This means that only attacks produced in an unintentional manner were considered. However, it was then pointed out that other attacks could be envisaged, such as the *sensitive attack* [4], in which the knowledge of the watermarking system could be exploited to erase the mark. In fact, in [4], attacks were classified between signal transformations and intentional attacks. With this classification, there are different approaches to define watermarking security [13,10,1,3], but there is no consensus at this time about this issue.

The main problem is whether the set of robustness attacks and the set of security attacks are disjoint or not and, then, some robustness attacks can be considered also as security attacks. Such a situation arises when we consider the definition proposed in [10], where the difference between robustness and security is defined in terms of the intentionality of the attack. Clearly, with this definition, the intersection between robustness and security attacks is not empty, since their classification only depends on the intention but not on the attack itself. However, some type of attacks can be uniquely classified using the distinction based on intentionality. For instance, any attack which exploits the knowledge of the watermarking embedding or detecting algorithm is intentional and can be labeled as a security attack. From a security point of view, the best strategy to protect any secure system against attacks which exploit the knowledge of the scheme's construction is to ensure the Kerckhoffs' principle [14]. Such principle establishes that the security of any system (cryptosystems in particular) must only depend on a secret key whereas all the other information concerning the system is public.

This paper is organised as follows. Section 2 describes the audio watermarking scheme. In Section 3, the ad-hoc attack for the watermarking scheme is presented and the modification to overcome this attack is suggested. Section 4 presents the performance of the modified scheme in terms of imperceptibility, capacity, robustness and security. Finally, Section 5 summarises the conclusions and the future research.

## 2 Audio Watermarking Scheme

The watermarking scheme (referred to as **Watermarking of Audio Content, WAUC**) presented in [18] (and improved in [19]) is described in the following sections.

### 2.1 Mark Embedding

Let the signal  $S$  to be marked be a collection of PCM samples. If the signal to be marked is stereo:  $S_{\text{stereo}} = [S_{\text{left}}, S_{\text{right}}]$ , both channels (left and right) must be added into a new "working" signal  $S = S_{\text{left}} + S_{\text{right}}$ . In the case of a mono signal, this step is not

required. The spectrum of  $S$ , denoted as  $S_F$ , is computed with a Fast Fourier Transform (FFT) algorithm. Then, the signal  $S$  is compressed using an MP3 algorithm with a rate of  $R$  kbps and decompressed again to PCM format. The result of this compression/decompression operation is a new signal  $S'$ , and its spectrum  $S'_F$  is obtained<sup>1</sup>. In the stereo case, the modified signal  $S'$  is obtained by adding the  $S'_{\text{left}}$  and  $S'_{\text{right}}$  which result after the compression and decompression operation.

Now, the set of marking frequencies  $F_{\text{mark}}$  is chosen as follows. Firstly, all  $f_{\text{mark}} \in F_{\text{mark}}$  must belong to the relevant frequencies  $F_{\text{rel}}$  of the original signal  $S_F$ :

$$F_{\text{rel}} = \{f \in [0, f_{\text{max}}] : |S_F(f)| \geq (p/100) |S_F|_{\text{max}}\}, \quad (1)$$

where  $f_{\text{max}}$  denotes the maximum frequency of the spectrum, which depends on the sampling rate and the sampling theorem<sup>2</sup>,  $p \in [0, 100]$  is a percentage and  $|S_F|_{\text{max}}$  is the maximum magnitude of the spectrum  $S_F$ . Note that the spectrum values in the interval  $[-f_{\text{max}}, 0]$  are the complex-conjugate of those in  $[0, f_{\text{max}}]$ .

Secondly, the frequencies to be marked are those for which the magnitude remains “unchanged” after lossy compression and decompression, where “unchanged” means a relative error below a given threshold  $\varepsilon$ :

$$F_{\text{mark}} = \{f_1, f_2, \dots, f_n\} = \{f \in F_{\text{rel}} : |(S_F(f) - S'_F(f)) / S_F(f)| < \varepsilon\}. \quad (2)$$

Similarly as done in the image watermarking scheme of [8], a 70-bit stream mark,  $W$  ( $|W| = 70$ ), is firstly extended to a 434-bit stream  $W_{\text{ECC}}$  ( $|W_{\text{ECC}}| = 434$ ) using a dual Hamming Error Correcting Code (ECC). This coding makes it possible to apply the watermarking scheme as a fingerprinting scheme robust against collusion of two buyers [7]. Finally, a pseudo-random binary stream (PRBS), generated with a cryptographic key  $k$ , is added to the extended mark as it is embedded into the original signal.

Once the frequencies in  $F_{\text{mark}}$  have been chosen, the spectrum of the marked signal is computed as:

$$\hat{S}_F(f) = \begin{cases} S_F(f), & f \notin F_{\text{mark}}, \\ S_F(f) \cdot 10^{\pm d/20} & f \in F_{\text{mark}}, \text{ to embed '1' (+d/20) or '0' (-d/20)}. \end{cases}$$

Since spectrum components in  $S_F$  are paired (pairs of complex-conjugate values), the same transformation (increase or decrease  $d$  dB) must be performed to  $S_F(f_{\text{mark}})$  and to its conjugate. In this process, the mark  $W_{\text{ECC}}$  is replicated as many times as required. In the stereo case, the magnitude modification step is applied to both  $S_{\text{left}}$  and  $S_{\text{right}}$  independently **at the same frequencies**. Finally, the marked audio signal is converted to the time domain  $\hat{S}$  applying an inverse FFT (IFFT) algorithm. As discussed in [18], this scheme has been designed to provide with “natural” robustness against lossy compression attacks.

## 2.2 Mark Detection

The objective of the mark detection algorithm is to determine whether an audio test signal  $T$  is a (possibly attacked) version of the marked signal  $\hat{S}$ . It is assumed that  $T$  is

<sup>1</sup> Throughout this paper, the Blade codec [12] (**coder/decoder**) for the MP3 algorithm has been chosen and, thus, the psychoacoustic model of this codec is implicitly used.

<sup>2</sup>  $f_{\text{max}} = \frac{1}{2T_s}$ , where  $T_s$  is the sampling time.

in PCM format or can be converted to it. Note that working signals adding the left and the right channels must be used in the stereo case.

First of all, the spectrum  $T_F$  is obtained applying the FFT algorithm and, then,  $|T_F(f_{\text{mark}})|$ , the magnitude at the marking frequencies, is computed for all  $f_{\text{mark}} \in F_{\text{mark}}$ . Note that this method is strictly positional and, because of this, it is required that the number of samples in  $\hat{S}$  and  $T$  is the same. If there is only a little difference in the number of samples, it is possible to complete the sequences with zeroes.

When the magnitudes  $|T_F(f_{\text{mark}})|$  are available, a scaling (Least Squares) step can be undertaken in order to minimize the distance between the sequences  $\lambda |T_F(f_{\text{mark}})|$  and  $|\hat{S}_F(f_{\text{mark}})|$  (see [18] for details). This LS step implicitly uses the embedded mark (since  $S_F(f_{\text{mark}})$  is needed) but it can be omitted ( $\lambda = 1$ ) or performed with the original signal  $S_F(f_{\text{mark}})$  instead of the marked one  $\hat{S}(f_{\text{mark}})$ .

Now, the ratios  $r_i = \lambda |T_F(f_i)| / |S_F(f_i)|$ , are computed to decide whether a ‘0’, a ‘1’ or a ‘\*’ (not identified) might be embedded at the  $i$ -th position. Given the interval

$$I = \left[ 10^{\frac{q}{20}}(100 - q)/100, 10^{\frac{q}{20}}(100 + q)/100 \right],$$

if  $r_i \in I \Rightarrow \hat{b}_i := \text{‘1’}$ , if  $1/r_i \in I \Rightarrow \hat{b}_i := \text{‘0’}$  and, otherwise,  $\hat{b}_i := \text{‘*’}$ . Here,  $q \in [0, 100]$  is a percentage and  $\hat{b}_i$  is the  $i$ -th component of the vector  $\hat{\mathbf{b}}$  which contains a sequence of “detected bits”. Finally, the PRBS signal is removed from the bits  $\hat{\mathbf{b}}$  to recover the true embedded bits  $\mathbf{b}$ . This operation must preserve unaltered the ‘\*’ marks.

Once  $\mathbf{b}$  has been obtained, its length  $n$  will be greater than the length of the extended mark. Hence, each bit of the mark appears at different positions in  $\mathbf{b}$ . A voting scheme (see [18] for details) is applied to choose whether the  $i$ -th bit of the mark is ‘1’, ‘0’ or unidentified (\*). As a result of this voting scheme, an identified extended mark  $W'_{\text{ECC}}$  is obtained and the error correcting algorithm is used to recover an identified 70-bit stream mark,  $W'$ , which will be compared with the true mark  $W$ .

The suggested scheme is informed (not blind) since the original signal is needed by the mark detection process. However, the bit sequence which forms the embedded mark is not required for detection (if the LS step is omitted or performed using  $S_F$ ), which makes this method suitable also for fingerprinting [2].

### 3 Security Issues

In this section, we focus on attacks which exploit the knowledge of the embedding and detection algorithms. More specifically, we consider the case referred to as *Watermark Only Attack*, in which the attacker has only access to marked contents [5,1]. An ad-hoc strategy can be specifically defined for the WAUC watermarking scheme once the mark embedding and detection algorithms are disclosed.

Concerning the WAUC watermarking scheme presented in the previous section, the disclosure of the mark embedding and detection algorithms has an obvious drawback from a security point of view: given the embedding parameters  $R$ ,  $\varepsilon$  and  $p$ , and the MP3 encoder/decoder, the position of the embedded bits ( $F_{\text{mark}}$ ) is absolutely determined. Therefore, a malicious attacker (Mallory) with knowledge about the embedding pro-

cess, could design an ad-hoc attack to disturb the spectrum of  $\hat{S}$  at those frequencies and try to erase the mark. The following section presents such attack.

### 3.1 Ad-Hoc Security Attack

Assuming that Mallory knows the embedding and detection algorithms, an ad-hoc security attack for this watermarking scheme can be described in the following way:

1. Mallory obtains the marked signal  $\hat{S}$ .
2. Mallory computes the spectrum  $\hat{S}_F$  applying the FFT.
3. Mallory encodes/decodes the marked signal  $\hat{S}$  with an MP3 encoder/decoder and gets a modified signal  $\hat{S}'$ . Here, he uses the bit rate  $R'$  for the MP3 encoder/decoder. Now, he applies the FFT to the signal  $\hat{S}'$  and gets the spectrum  $\hat{S}'_F$ .
4. Mallory computes the set  $\hat{F}_{\text{mark}}$  applying the criteria of Equations 1 and 2 using the spectra  $\hat{S}'_F$  instead of  $S_F$ , using  $\hat{S}'_F$  instead of  $S'_F$ , and the parameters  $p'$  and  $\varepsilon'$ . Note that Mallory does not have the original signal  $S$  neither the modified signal  $S'$ .
5. Finally, Mallory disturbs the magnitude of the spectrum at the frequencies.  $\hat{F}_{\text{mark}}$ . For example, he could decide to disturb  $\pm d'$  dB at those frequencies randomly.

Note that, even if Mallory knew the mark  $W$  and the extended mark  $W_{\text{ECC}}$ , the use of the PRBS generated with a **secret** key  $k$  in the embedding process prevents Mallory from knowing which exact bit is embedded at each position. So, even if Mallory got the exact set  $F_{\text{mark}}$  (which is impossible unless he had the original signal), he would not know whether if a binary '0' or a binary '1' is embedded at each position. Therefore, the best strategy to disturb the marked signal is to add or subtract  $d'$  dB randomly.

Of course, Mallory should gain knowledge of the embedding parameters in order to have all the information required to construct his approximation to  $F_{\text{mark}}$ . If the parameters  $R$ ,  $p$  and  $\varepsilon$  were public, the attack would be easier, since Mallory could set  $R' = R$ ,  $p' = p$  and  $\varepsilon' = \varepsilon$ . In fact, the sets  $F_{\text{mark}}$  and  $\hat{F}_{\text{mark}}$  cannot be exactly the same, since Mallory should have access to the original signal  $S$  to obtain  $F_{\text{mark}}$ . However, as  $\hat{S}$  is expected to be quite similar to  $S$  (since the WAUC scheme has a good imperceptibility level), the constructed set  $\hat{F}_{\text{mark}}$  will contain many of the frequencies in the original  $F_{\text{mark}}$ , possibly enough to be able to delete the mark.

In addition, it must be noticed that although Mallory does not know the exact values of the embedding parameters, he can construct an approximate superset  $\hat{F}_{\text{mark}}$  following these guidelines:

1. Choose a large enough parameter  $R'$ . The larger  $R'$  is, the more similar  $\hat{S}$  and  $\hat{S}'$  become and, thus, the more frequencies will be included in  $\hat{F}_{\text{mark}}$ .
2. Choose a small enough percentage  $p'$ . This way, more frequencies satisfy the criterion of Equation 1.
3. Choose a large enough relative error  $\varepsilon'$ . This way, more frequencies satisfy the criterion of Equation 2.

Here, the parameter  $d'$  should be chosen in such a way that the binary '1's and '0's are erased. Thus, an advisable choice for  $d'$  is  $d' > d$ . Since the imperceptibility of the

mark requires that  $d$  is not very large [20], usually  $d \leq 1$  dB will be chosen for mark embedding. Hence, Mallory decides to use  $d' = 2$  dB. As a consequence of Mallory's attack, the perceptual quality of  $T$  will be reduced with respect to that of  $\hat{S}$ , but this is the price an attacker has to pay in order to delete the mark.

### 3.2 Security Enhancement

The main idea to solve the security problem described above is to “hide” the marking positions as much as possible by computing them using a secret key, as detailed below. Then, the watermarking scheme can be considered secure under the Kerckhoffs' assumption. On the other hand, such modification should preserve as many properties as possible compared to the original “non-secure” scheme. Special attention should be devoted to robustness, imperceptibility and capacity.

If these two conditions are met, it would be possible to make the watermarking scheme publicly known except for the secret key. The unavailability of the key should make it very difficult to proceed with the attack presented in section 3.1. In order to meet these conditions, the embedding process can be modified in such a way that the marking frequencies depend on a secret key as follows:

- Proceed with the mark embedding process described in Section 2.1 until the set  $F_{\text{mark}}$  is obtained. Now, define  $f_M = \max F_{\text{rel}}$  (the maximum frequency that satisfies the criterion of Equation 1) and  $F_{\text{perc}} = F_{\text{mark}}$ . The set  $F_{\text{mark}}$  obtained in Section 2.1 is a temporary variable ( $F_{\text{perc}}$ ) which stores the most perceptually relevant part of the spectrum, hence the subindex *perc*.
- Define the set of candidate marking frequencies as  $F_{\text{cand}} = [0, f_M]$ . This prevents very high frequencies, which are not usually good for embedding the mark as robustness is concerned, to be chosen. Let  $m$  be the cardinality of the set  $F_{\text{cand}}$ , *i.e.*  $m = |F_{\text{cand}}|$ . Note that  $F_{\text{perc}} \subseteq F_{\text{cand}}$ <sup>3</sup>.
- Choose a **pseudo-random** number generator in the range  $[0, 1]$  and a **secret key**  $k_{\text{sec}}$  as the (initial) seed<sup>4</sup>.
- Choose two probabilities  $p_1, p_2 \in [0, 1]$  such that a given frequency which belongs to the set  $F_{\text{perc}}$  will be chosen for marking with a probability  $p_1$ , and  $p_2$  is the probability of choosing a frequency in the set  $F_{\text{cand}} - F_{\text{perc}}$ , where “ $-$ ” stands for the set subtraction operation.
- Reset the random number generator with the seed  $k_{\text{sec}}$ . Let  $F_{\text{mark}}$  be the empty set, and proceed as follows. For all the frequencies  $f$  in the set  $F_{\text{cand}}$  do:
  1. Generate a random number  $r \in [0, 1]$ .
  2. If ( $f \in F_{\text{perc}}$  **and**  $r < p_1$ ) **or** ( $f \notin F_{\text{perc}}$  **and**  $r < p_2$ ) then  $F_{\text{mark}} := F_{\text{mark}} \cup \{f\}$  (the frequency  $f$  is included into the set of marking frequencies).
  3. Otherwise, discard the frequency  $f$ .

Once the set  $F_{\text{mark}}$  has been generated, it is possible to apply the mark embedding process presented in Section 2.1 with this new set. The mark reconstruction process

<sup>3</sup> Usually, the set  $F_{\text{cand}}$  has many more elements than  $F_{\text{perc}}$ , *i.e.*  $|F_{\text{perc}}| \ll |F_{\text{cand}}|$ .

<sup>4</sup> The subindex *sec* is used to distinguish from the secret key  $k$  which was already used in the embedding process described in Section 2.1.

should be also modified accordingly, since the mark embedding detection must repeat the first few steps of the mark embedding process in order to obtain  $F_{\text{mark}}$  (and  $\hat{S}$ ).

It is worth pointing out some remarks about the modification suggested above. Firstly, note that the frequencies in  $F_{\text{perc}}$  are included into the set  $F_{\text{mark}}$  with a probability  $p_1$  and those in  $F_{\text{cand}} - F_{\text{perc}}$  are included with a probability  $p_2$ . Secondly, any pseudo-random number generator can be used and, thus, the length of the secret key  $k_{\text{sec}}$  will depend on it. In this paper, the Mersenne Twist method presented in [17] is used. In addition, in the original watermarking scheme presented in Section 2, the number of marked bits is  $n = |F_{\text{perc}}|$  (remember that  $F_{\text{mark}}$  is now referred to as  $F_{\text{perc}}$ ). It is possible to define  $p_1$  and  $p_2$  such that the expected value of the final number of elements in  $F_{\text{mark}}$  is the same as in the scheme presented in Section 2. The expected value of the number of elements in  $F_{\text{mark}}$  is the following:  $E(|F_{\text{mark}}|) = np_1 + (m - n)p_2$ . In order to get  $E(|F_{\text{mark}}|) = n$ ,  $p_2$  should be chosen as:

$$np_1 + (m - n)p_2 = n \Leftrightarrow p_2 = (1 - p_1) \frac{n}{m - n},$$

which is considered as the **default value** for  $p_2$  hereafter. With this default value for  $p_2$ , the number of marking frequencies  $n' = |F_{\text{mark}}|$  will be (in average) equal to  $n$  (since  $E(n') = n$ ). In addition, the ratios of frequencies in  $F_{\text{mark}}$  belonging to  $F_{\text{perc}}$  and  $F_{\text{cand}} - F_{\text{perc}}$  will be (in average)  $p_1$  and  $1 - p_1$ , respectively. Thus,  $p_1$  determines the balance between the marking frequencies inside and outside  $F_{\text{perc}}$ . For example, with  $p_1 = 0.2$  (and  $p_2$  equal to the default value), a 20% (in average) of the frequencies in  $F_{\text{mark}}$  belong to the set  $F_{\text{perc}}$  and the other 80% belong to  $F_{\text{cand}} - F_{\text{perc}}$ .

Note, also, that if  $p_1 = 1$  and the default value is chosen for  $p_2$ , then  $p_2 = 0$ . In this case, the set  $F_{\text{mark}}$  becomes identical to  $F_{\text{perc}}$  and the modified watermarking scheme becomes identical to the one presented in [19].

The robustness of the modified scheme will depend on the value of  $p_1$ . *A priori*, the frequencies in the set  $F_{\text{perc}}$  are better for mark embedding, since they have been chosen in such a way that MP3 compression attacks can be overcome [19]. It is expected that  $p_1 = 1$  is the best value for robustness, but such a value is not advisable for security. Thus, a trade-off solution between robustness and security must be attained. As imperceptibility is concerned, the frequencies  $F_{\text{perc}}$  refer to the most perceptible part of the spectrum and, thus, the lower  $p_1$  is, the better imperceptibility is expected.

One may think that Mallory could distort not only the frequencies in the set  $F_{\text{perc}}$ , but all the frequencies in the set  $F_{\text{cand}}$ . Note that, if it were the case, the mark would very probably be erased, but the distortion introduced to the signal would be so large (the spectrum would be distorted at all the low frequencies) that the attacked signal would be unusable in most typical situations.

Last, but not least, note that  $p_1$  should not be chosen too small or, at least, Mallory should not have knowledge about  $p_1$ . If  $p_1$  is so small that most of the marking frequencies lie outside the set  $F_{\text{perc}}$ , then Mallory would do better attacking at the frequencies in  $F_{\text{cand}} - \hat{F}_{\text{perc}}$ . If he attacked at those frequencies, he would be able to delete more bits and, in addition, he would disturb the least perceptible part of the spectrum resulting in an attacked signal with very good audio quality.

The secret key  $K$  of the modified watermarking scheme in order to increase security should be formed, at least, by the pseudo-random seeds  $k$  and  $k_{\text{sec}}$ , and the

probabilities  $p_1$  and  $p_2$ . Here, we consider that the secret key is the following:  $K = \{R, p, \varepsilon, d, k, k_{\text{sec}}, p_1, p_2\}$ , which is required for both mark embedding and detection. The decision whether  $R, p, \varepsilon$  and  $d$  are part of the secret key or public values can depend on the application. In any case, these parameters should not be used to enhance security but rather for tuning reasons [20]. Note, also, that the parameter  $q$  does not affect the mark embedding process. Thus,  $q$  is not a part of the secret key. Hereafter, the modified scheme is referred to as WAUC-sec.

## 4 Performance Evaluation

In this section, the WAUC and the WAUC-sec schemes are evaluated in terms of capacity, imperceptibility, robustness and security. Both the WAUC and the WAUC-sec schemes have been implemented using a dual binary Hamming code  $DH(31, 5)$  as ECC and the PRBS has been generated with the a DES cryptosystem in an Output Feedback (OFB) mode. A 70-bit mark  $W$  ( $|W_{\text{ECC}}| = 434$ ) was embedded. In addition, the following values have been chosen for the embedding and detection parameters:  $R = 128$  kbps,  $p = 2$ ,  $\varepsilon = 0.05$ ,  $d = 1$  dB and  $q = 10$ .

To test the performance of the audio watermarking schemes described in the previous sections, different audio files provided in the Sound Quality Assessment Material (SQAM) page [9] have been used. In order to summarise the results as much as possible, only the experiments performed for the (stereo) violoncello (melodious phase) file<sup>5</sup> are shown. Completely analogous results have been obtained for the other files in the SQAM corpus set. The properties of the WAUC-sec scheme have been tested for nine values of the probability  $p_1 \in \{0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1\}$  and the default value for  $p_2$ . In addition, eight different values of the secret key  $k_{\text{sec}}$  have been chosen randomly. Different values of  $k_{\text{sec}}$  are used in order to avoid biased results which could arise with some sequences of pseudo-random numbers. Note that the original WAUC scheme is also considered, since it is identical to WAUC-sec with  $p_1 = 1$  and  $p_2 = 0$ .

### 4.1 Capacity

*Capacity* ( $C$ ) is the amount of information that may be embedded and recovered in the audio stream and it is measured in bits per second (bps). It must be taken into account that the true capacity  $C$  is not the number of marked bits  $n'$  (the size of the set  $F_{\text{mark}}$ ). Hence,  $C = 70 \cdot n' / (434 \cdot l)$ , where  $l$  is the length of the marked signal.

Table 1 shows the capacity results obtained for the WAUC (the column with  $p_1 = 1$ ) and the WAUC-sec ( $0 \leq p_1 \leq 0.875$ ) schemes. Since eight different values of the key  $k_{\text{sec}}$  have been chosen, three different measures are shown in the table: the maximum, the minimum and the average. It can be observed that all the values are very similar, as the modification has been designed in such a way that the capacity of the original scheme is preserved when the default value is chosen for  $p_2$ . If more than eight values of the secret key  $k_{\text{sec}}$  had been used, the average value of  $C$  in each column would be

<sup>5</sup> In fact, only the first ten seconds ( $441000 \times 2$  samples) of the file have been taken into account.

**Table 1.** Capacity results for WAUC and WAUC-sec

Capacity	WAUC-sec								WAUC $p_1 = 1$
	$p_1$								
	0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	
Maximum $C$ (bps)	62.56	62.42	61.68	61.39	61.76	61.69	62.05	61.76	61.08
Minimum $C$ (bps)	59.06	58.60	59.32	59.31	59.47	59.55	59.73	60.34	61.08
Average $C$ (bps)	60.93	60.86	60.70	60.46	60.64	60.68	60.85	61.07	61.08

closer to that of the original scheme (61.08 bps). It must be taken into account that capacity also depends on the original signal, as discussed in [19].

## 4.2 Imperceptibility

*Imperceptibility* is concerned with the audio quality of the marked signal  $\hat{S}$  with respect to  $S$ . Here, to measure such property, the Objective Difference Grade (ODG) based on the ITU-R Recommendation standard BS.1387 [11] and the signal-to-noise ratio (SNR) are used. The BS.1287 standard is used for Perceptual Evaluation of Audio Quality (PEAQ) [21]. In particular, the implementation provided in the tool EAQUAL [16] has been used in this paper. The computed ODG values are in the range  $[-4, 0]$ , where 0 means imperceptible,  $-1$  means perceptible but not annoying,  $-2$  means slightly annoying,  $-3$  means annoying and  $-4$  means very annoying. The SNR values make it possible to compare these results with those presented in previous papers [19,20].

**Table 2.** Imperceptibility results for WAUC and WAUC-sec

Imperceptibility	WAUC-sec								WAUC $p_1 = 1$
	$p_1$								
	0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	
Maximum ODG	-0.46	-0.88	-1.17	-1.31	-1.53	-1.72	-1.81	-1.85	-1.96
Minimum ODG	-0.53	-0.96	-1.30	-1.58	-1.70	-1.80	-1.92	-2.01	-1.96
Average ODG	-0.50	-0.92	-1.24	-1.45	-1.64	-1.77	-1.87	-1.95	-1.96
Maximum SNR (dB)	39.76	28.58	25.77	23.95	22.23	21.15	20.42	19.71	18.95
Minimum SNR (dB)	39.18	26.69	24.50	23.13	21.86	20.75	19.94	19.47	18.95
Average SNR (dB)	39.47	27.85	25.10	23.49	22.00	20.96	20.19	19.55	18.95

In Table 2, the SNR and ODG measures obtained with both WAUC and WAUC-sec are shown. It is observed that the ODG values increase from imperceptible for  $p_1 = 0$  to slightly annoying for  $p_1 = 1$ , and analogous results are obtained in terms of SNR. These results are quite satisfying in general. For values of  $p_1 \leq 0.5$ , a listener is not expected to notice any remarkable difference between the original and the marked files, since the ODG values range between imperceptible and perceptible but not annoying.

Imperceptibility has been improved with respect to the original scheme. Thus, the modifications do not only enhance security, but also imperceptibility. These results were expectable, since the changes introduced to the scheme decrease the number of relevant frequencies at which the magnitude is disturbed. Finally, note that the imperceptibility results might be further improved by tuning the embedding parameters carefully [20].



### 4.3 Robustness

*Robustness* is the resistance to accidental removal of the embedded bits. The robustness of the resulting scheme has been tested against the version 0.2 of the StirMark Benchmark for Audio (SMBA) [6] and also against MP3 compression. In particular, 43 different attacks of the SMBA have been tested, since the attacks which modify the number of samples in a significant way cannot be tested with the current version of the scheme. Robustness has been assessed using a correlation measure between the embedded and the identified marks ( $W$  and  $W'$ ):

$$\text{Correlation} = \frac{1}{|W|} \sum_{i=1}^{|W|} \beta_i.$$

where  $\beta_i = 1$  if  $W_i = W'_i$  and  $-1$  otherwise. In this paper, we consider that the watermarking scheme survives an attack if the Correlation  $\geq 0.8$ .

**Table 3.** Robustness results against the SMBA for WAUC and WAUC-sec

		WAUC-sec							WAUC
Robustness	$p_1$								
	0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	$p_1 = 1$
SMBA test	34/43	35/43	36/43	36/43	36/43	37/43	37/43	35/43	35/43

The robustness of WAUC and WAUC-sec are not very different with respect to the SMBA. Table 3 shows the robustness results obtained for the WAUC and WAUC-sec schemes with a value for the key  $k_{sec}$  (analogous results have been obtained with other keys). The values in the table are given in a  $x/43$  ratio meaning how many SMBA attacks out of the 43 attacks performed have been survived. It can be noticed that the survival ratio varies from 34/43 to 37/43 but there is not a monotonic pattern for this variation. This result is not very surprising since the SMBA attacks are not specifically designed to disturb the most perceptually significant frequencies of the spectrum.

However, the robustness of the WAUC and WAUC-sec schemes against MP3 compression attacks are quite different. Table 4 summarises the robustness results against MP3 compression using all eight random values for the key  $k_{sec}$ . The attacks have been performed with a Blade codec and all the allowed bit rates: 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320 kbps. The scores given in the table are in the form: Score = # Survived attacks/# Performed attacks. Therefore, scores of 1 mean that the scheme survives the compression attack for all the values of the key  $k_{sec}$ , whereas 0 means that the compression attack has successfully erased the mark for all values of  $k_{sec}$ . In the WAUC scheme (which does not depend on the key  $k_{sec}$ ), the only possible scores are 0 and 1, whereas intermediate results are possible for the WAUC-sec scheme. As it can be observed, both schemes are able to overcome MP3 compression attacks for all bit rates of 128 kbps and higher. The robustness against the other bit rates increases as  $p_1$  is larger, as expected, since more marking frequencies belong to the most perceptually significant part of spectrum.

**Table 4.** Robustness results against MP3 compression attacks for WAUC and WAUC-sec

MP3 bit rate	WAUC-sec								WAUC
	$p_1$								$p_1 = 1$
	0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	
32 kbps	0	0	0	0	0	0	0	0	0
40 kbps	0	0	0	0	0	0	0	0.25	0
48 kbps	0	0	0	0	0	0	0.375	0.625	1
56 kbps	0	0	0	0	0.125	0.5	0.625	1	1
64 kbps	0	0	0	0.125	0.75	1	1	1	1
80 kbps	0	0	0.5	1	1	1	1	1	1
96 kbps	0.125	1	1	1	1	1	1	1	1
112 kbps	0.375	1	1	1	1	1	1	1	1
[128, 320] kbps	1	1	1	1	1	1	1	1	1

**Table 5.** Security results for WAUC and WAUC-sec

		WAUC-sec		WAUC
		$p_1$		$p_1 = 1$
		0.25	0.5	
Attack #1	Correlation	$1 > 0.8 (*)$	$0.9077 > 0.8 (*)$	$-0.0462 < 0.8$
	ODG	$-2.1845$	$-2.0159$	$-1.6894$
Attack #2	Correlation	$1 > 0.8 (*)$	$0.6308 < 0.8$	$0.2308 < 0.8$
	ODG	$-2.4946$	$-2.3492$	$-2.1868$

As MP3 compression is concerned, the WAUC scheme performs better than WAUC-sec, since it survives attacks for all bit rates greater to or equal to 48 kbps. The WAUC-sec scheme is not that robust, but it still produces good enough results for many values of  $p_1$ . For example, with  $p_1 = 0.5$  the WAUC-sec scheme is able to overcome all the attacks with bit rates greater to or equal to 80 kbps, and a 75% of the attacks performed with 64 kbps. If high quality is required (for example for music audio files), it is not expected that an attacker would use MP3 bit rates lower than 80 kbps.

### 4.4 Security

In this section, two different kinds of experiments are presented. Firstly, false positive results are shown and, secondly, the resistance of both WAUC and WAUC-sec against the ad-hoc attack presented in section 3.1 is examined.

As false positives are concerned, the experiments consist of using different values for the key  $k_{sec}$  in the embedding and the detection processes. For these experiments, the value  $p_1 = 0.5$  has been chosen. The correlation measure described in the previous section has been computed to assess the similarity between the embedded mark  $W$  and the recovered one  $W'$ . Since eight different values have been used for  $k_{sec}$ , seven correlation values are obtained for each  $k_{sec}$ . Thus,  $8 \times 7 = 56$  experiments have been performed. If any of these 56 correlation values were too close to 1, then the false positive rate would be relatively large. In these 56 experiments, the *maximum* correlation value obtained is 0.3231, quite far from the required survival threshold (0.8). The average of the absolute values of all these 56 correlation measures is lower than 0.1.

The results for two settings of the ad-hoc attack depicted in Section 3.1 are given below. **Attack #1:** performed assuming that Mallory correctly guesses the parameters

used by the embedder:  $R' = R = 128$  kbps,  $p' = p = 2$  and  $\varepsilon' = \varepsilon = 0.05$ . The magnitude modification parameter  $d' = 2$  dB has been chosen. **Attack #2**: it is assumed that Mallory does not know the parameters used by the embedder and he tries to produce an approximate superset of  $F_{\text{perc}}$  by choosing:  $R' = R = 128$  kbps,  $p' = 1$  ( $p = 2$ ) and  $\varepsilon' = 0.1$  ( $\varepsilon = 0.05$ ). Again,  $d' = 2$  dB is used. This attack will affect more frequencies than the previous one.

For these attacks, the LS step mentioned in Section 2.2 has not been used ( $\lambda = 1$ ). The experiments have been carried out for two values of  $p_1$  (0.25 and 0.5) and the same secret key  $k_{\text{sec}}$ . The results obtained for this ad-hoc attack are summarised in Table 5, where the ODG results do not refer to the original (unmarked) file but to the marked signal. The ODG values with respect to the original file are obviously worse. The star sign “(\*)” shows which attacks are not successful, *i.e.* when the correlation is larger than or equal to the threshold 0.8. The WAUC-sec scheme is able to survive Attack #1 for both values of  $p_1$ , whereas the original WAUC scheme fails to recover the mark. Note, also that the ODG values are worse than those obtained for mark embedding (see Table 2). Thus the WAUC-sec scheme survives attacks which introduce more distortion into the attacked signal than what the embedder does in the marking process. The situation is a bit more difficult with the Attack #2, since the spectrum is disturbed at more frequencies. Because of this, the ODG values are worse than those of the Attack #1. Now the WAUC-sec scheme does not survive the attack with  $p_1 = 0.5$ , though the correlation value is not too far from the threshold. These results show that the modification works exactly as predicted in Section 3, since the ad-hoc attack fails to erase the mark as the probability  $p_1$  is decreased.

Two final experiments have been performed to test a worse attack scenario. Firstly, it is assumed that Mallory wants to disturb not only the frequencies in the set  $F_{\text{perc}}$ , but all the frequencies in  $F_{\text{cand}}$ . Such attack successfully erases the mark even for  $p_1 = 0.25$ , since the obtained correlation is  $-0.1692$ . However, in such a situation, the ODG value between the marked file and the attacked one is  $-2.6933$ , *i.e.* the noise introduced by the attack is quite annoying according to the ODG scale defined above. Secondly, if Mallory guesses (or discovers) that  $p_1 = 0.25$  has been chosen, he would do better attacking  $F_{\text{cand}} - F_{\text{perc}}$ , as already remarked in Section 3.2. Such attack has been performed with  $R' = R = 128$  kbps,  $p' = p = 2$ ,  $\varepsilon' = \varepsilon = 0.05$  and  $d' = 2$  dB, disturbing the spectrum at the frequencies  $F_{\text{cand}} - \hat{F}_{\text{perc}}$ . In this case, the correlation is 0.3538, but audio quality is very good according to the ODG measure:  $-1.2195$ .

The main conclusion after all these experiments is that the value of  $p_1$  should be chosen in some interval centred at 0.5 and too small values should be avoided. For example,  $p_1$  should be in the interval  $[0.3, 0.7]$ . This way, a good trade-off between robustness and security would be obtained.

## 5 Conclusions

In this paper, several security issues related to a watermarking scheme for audio (WAUC) are discussed. On the first hand, it has been shown that the original WAUC scheme is not suitable for the disclosure of the embedding and detection algorithms, since the marking positions could be exposed to a malicious user, making it possible

to design a successful ad-hoc attack. A modification, WAUC-sec, has been described in such a way that the embedding and the detection processes depend on a secret key without which the marking positions cannot be exactly determined.

The experiments show that it is possible to tune the modification so that the capacity of the original scheme can be preserved. In addition, the WAUC-sec scheme obtains better imperceptibility results (both in ODG and SNR measures) than the original counterpart for the same capacity. As robustness is concerned, both schemes produce similar results against the SMBA, but the original WAUC scheme is more robust against MP3 compression. Finally, concerning security, both false positive experiments and ad-hoc attacks have been performed. On the one hand, it has been shown that false positives are quite improbable if different secret keys are used for embedding and detection. On the other hand, the ad-hoc attacks can be survived by the WAUC-sec scheme whereas they successfully erase the mark when the original WAUC scheme is used. In short, the WAUC-sec scheme provides with a trade-off solution between security and robustness against MP3 compression.

There are several directions to further the research presented in this paper. The first one is to take into account some attacks which are not included in the version 0.2 of the SMBA, such as play speed variance attacks. Secondly, the development of a real application would require working with frames (blocks of samples) instead of the whole file, which would imply some reformulation. Finally, the possibility of obtaining a blind detector should be investigated.

## Acknowledgements and Disclaimer

This work is partially supported by the Spanish MCYT and the FEDER funds under grants TIC2003-08604-C04-04 MULTIMARK and SEG2004-04352-C04-04 PROPRIETAS-WIRELESS. The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

## References

1. M. Barni, F. Bartolini, and T. Furon. A general framework for robust watermarking security. *Signal Process.*, 83(10):2069–2084, 2003.
2. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. In *Advances in Cryptology-CRYPTO'95*, LNCS 963, pages 452–465. Springer-Verlag, 1995.
3. F. Cayre, C. Fontaine, and T. Furon. Watermarking attack: Security of wss techniques. In *Digital Watermarking: Third International Workshop*, volume LNCS 3304, pages 197–208, Seoul, Korea, Oct 2004.
4. I. Cox and J.-P. Linnartz. Some general methods for tampering with watermarks. *IEEE Journal on Selected Areas in Communications*, 16:587–593, May 1998.
5. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, 22(6):644–654, November 1976.

6. J. Dittman, M. Steinebach, A. Lang, and S. Zmudzinski. Advanced audio watermarking benchmarking. In *Proceedings of the IS&T/SPIE's 16th Annual Symposium on Electronic Imaging*, volume 5306 - Security, Steganography, and Watermarking of Multimedia Contents VI, Sant Jose, CA, US, January 2004.
7. J. Domingo-Ferrer and J. Herrera-Joancomartí. Short collusion-secure fingerprinting based on dual binary hamming codes. *Electronics Letters*, 36(20):1697–1699, September 2000.
8. J. Domingo-Ferrer and J. Herrera-Joancomartí. Simple collusion-secure fingerprinting schemes for images. In *Proceedings of the Information Technology: Coding and Computing ITCC'2000*, pages 128–132. IEEE Computer Society, 2000.
9. EBU. SQAM - Sound Quality Assessment Material, 2001. <http://sound.media.mit.edu/mpg4/audio/sqam/>.
10. T. Furon et al. Security analysis. Deliverable 5. 5, 2002. IST project CERTIMARK (IST-1999-10987).
11. ITU-R. Recommendation BS.1387. Method for objective measurements of perceived audio quality, December 1998.
12. T. Jansson. Homepage for BladeEnc, 2001. <http://bladeenc.mp3.no/>.
13. T. Kalker. Considerations on watermarking security. In *Proceedings of the IEEE Fourth Workshop on Multimedia Signal Processing*, pages 201–206, Cannes, France, Oct. 2001.
14. A. Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, 9:5–38, January 1883.
15. A. Lang, J. Dittmann, and E. J. Delp. Application-oriented audio watermark benchmark service. In *Proceedings of the IS&T/SPIE's 17th annual symposium on Electronic Imaging*, volume 5681, San Jose, CA, Jan. 2005.
16. A. Lerch. EAQUAL - Evaluate Audio QUALity. <http://www.mp3-tech.org/programmer/sources/eaqual.tgz>.
17. M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.
18. D. Megías, J. Herrera-Joancomartí, and J. Minguillón. A robust audio watermarking scheme based on MPEG 1 layer 3 compression. In *Communications and Multimedia Security - CMS 2003*, Lecture Notes in Computer Science 2828, pages 226–238, Turin (Italy), October 2003. Springer-Verlag.
19. D. Megías, J. Herrera-Joancomartí, and J. Minguillón. An audio watermarking scheme robust against stereo attacks. In *Proceedings of the Multimedia and Security Workshop*, pages 206–213, Magdeburg (Germany), September 2004. ACM.
20. D. Megías, J. Herrera-Joancomartí, and J. Minguillón. Robust frequency domain audio watermarking: a tuning analysis. In *International Workshop on Digital Watermarking - IWDW 2004*, Lecture Notes in Computer Science 3304, pages 244–258, Seoul (Korea), November 2004. Springer-Verlag.
21. T. Thiede, W. C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, C. Colomes, M. Keyhl, G. Stoll, K. Brandenburg, and B. Feiten. PEAQ – The ITU standard for objective measurement of perceived audio quality. *J. Audio Eng. Soc.*, 48(1-2):3–29, Jan.-Feb. 2000.

# Reversible Watermark with Large Capacity Using the Predictive Coding

Minoru Kuribayashi<sup>1</sup>, Masakatu Morii<sup>1</sup>, and Hatsukazu Tanaka<sup>2</sup>

<sup>1</sup> Department of Electrical and Electronics Engineering,  
Faculty of Engineering, Kobe University,  
1-1, Rokkodai-cho, Nada-ku, Kobe, 657-8501 Japan

{kminoru, mmorii}@kobe-u.ac.jp

<sup>2</sup> Kobe Institute of Computing,  
2-2-7, Kanou-cho, Chuo-ku, Kobe, 650-0001 Japan  
tanaka@kic.ac.jp

**Abstract.** A reversible watermarking algorithm with large capacity has been developed by applying the difference expansion of a generalized integer transform. In this algorithm, a watermark signal is inserted in the LSB of the difference values among pixels. In this paper, we apply the prediction errors calculated by a predictor in JPEG-LS for embedding a watermark signal, which contributes to increase the amount of embedded information with less degradation. As one of the drawbacks discovered in the above conventional method is a large size of the embedded location map introduced to make it reversible, we decrease the large size of the location map by vectorization, and then modify the composition of the map using the local characteristic in order to enhance the performance of JBIG2.

## 1 Introduction

In a data-hiding technique[1], the embedding causes irreversible degradation to an image. Although the degradation is perceptually slight, it may not be accepted to some applications such as medical or military images. For the countermeasure, lossless data-hiding techniques, which is called reversible(invertible) watermark, have been developed. The reversible watermark techniques might be classified into two methods. One compresses features of an image and transmits the compressed bit-stream as a part of the embedding information. At the decoding, the embedded information including the compressed bit-stream is extracted, and the original image is restored by replacing the modified features with the decompressed original features. In [2], each pixel is first quantized by a quantization step size  $L$ , and appends the embedding information to the compressed quantization noise. Then, the information is added to the quantized image. The scheme tends to be superior when the watermarked image keeps high quality. However, for a large amount of information, the capacity would be inadequate.

The other method uses reversible integer transforms to the spatial domain of an image, and embeds a watermark information in the transformed signal values.

Tian[3] presented a difference expansion transform of pair of pixels, which is haar wavelet transform, to devise a reversible watermark with a large capacity and low degradation. His algorithm divides an image into pairs of pixels, then it inserted one bit into the difference of the pixels of each pair from those pairs that are not expected to causes an overflow or underflow. So as to recover the original image, a location map that indicates the modified pairs is embedded as a part of the embedding information after compression. Heijmans et al.[4] improves the capacity for high quality images by performing low pass filtering to an image to predict a location map. Alattar[5] extends Tian's method to the difference expansion of a vector of several pixels to achieve larger capacity. The algorithm can insert several bits in the difference expansion of each vector of adjacent pixels. As each element in the location map indicates the embedding position of the corresponding pair/vector, the size of the map depends on the number of the pairs/vectors. Even if the location map is compressed, the size is still large, and hence the capacity is restricted in the above schemes. In addition, the positions where a watermark signal is embedded is mainly at a flat region of an image for the property of the algorithm, which causes perceptual degradation.

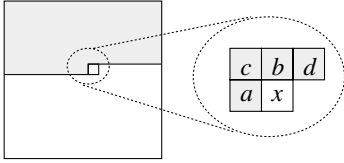
In this paper, we propose a new technique to embed a large amount of information with less degradation. Our main idea is to embed a watermark in prediction errors calculated by a well-designed predictor in JPEG-LS[6]. Since the average value of the prediction errors is small, our scheme can spread out the positions where a watermark signal can be embedded without causing both overflow and underflow. Such positions are not restricted only in a flat region, but also in a noisy one, and hence the distortions caused by our embedding may be less perceived. One of the drawbacks in the conventional schemes[3]-[5] is the size of the location map even if it is compressed. First, we reduce the size of the map itself by making a vector which treats several pixels as one cluster. And we study the characteristic of the local conditions of pixels, and find that at some pixels the location map is not necessary if a certain condition is satisfied. Such characteristic is exploited to enhance the performance of the compression algorithm. As the results, our scheme achieves very large capacity with less distortions. Since the operation is reversible, multiple embedding is possible and hence the capacity may be increased further.

## 2 Preliminary

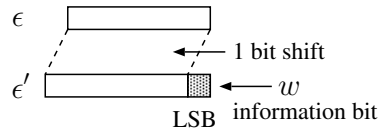
### 2.1 Modeling in JPEG-LS

JPEG-LS[6] is the algorithm at the core of the new ISO/ITU standard for lossless and near-lossless compression. The algorithm attains significantly better compression ratios, similar or superior to those obtained with state-of-the art schemes based on arithmetic coding, but at a fraction of the complexity.

Lossless image compression schemes, in general, consist of two distinct and independent components: *modeling* and *coding*. The modeling part is formulated as an inductive inference problem, in which an image is observed sample by sample in some predefined order. In JPEG-LS, the sample points are defined as



**Fig. 1.** The sample points of the fixed predictors



**Fig. 2.** Expanding of the prediction error to embed a watermark

$a, b, c,$  and  $d$  depicted in Fig.1. And the current sample  $x$  is predicted as  $\hat{x}$  using the four samples. In the coding part, the predictive error  $\epsilon$

$$\epsilon = x - \hat{x} \quad (1)$$

is encoded with an extended family of Golomb-type code[8] which is adaptive symbol-by-symbol coding at very low complexity.

The distribution of the prediction error  $\epsilon$  is well modeled by a *two-sided geometric distribution* (TSGD)[7] centered at zero. And  $\epsilon$  is also extremely sensitive for the changes in the previously occurred samples. Note that the change occurred in one sample is propagating to the following every prediction error.

## 2.2 Expanding

The main idea of our scheme is to utilize the prediction errors calculated by the predictor in JPEG-LS to embed information bits in an image. Since the prediction errors follow TSGD centered at zero, the average value might be small. Based on the characteristic, a lossless embedding could obtain a better performance than that of haar wavelet transform[3][4] and generalized integer transform[5]. In JPEG-LS, only the prediction errors are preserved assisted by entropy coding[8]. Therefore, if the original prediction error can be recovered from a watermarked image, the embedding operation is reversible.

When an information bit  $w$  is embedded, a prediction error  $\epsilon$  is expanded so as to insert  $w$  without loss of original information of  $\epsilon$ . The basic embedding operation is to double  $\epsilon$ , and to put  $w$  on its LSB(See Fig.2).

$$\epsilon' = 2\epsilon + w, \quad (2)$$

It is the same operation as that of the conventional schemes[3]-[5]. Those schemes calculate the difference among neighboring pixels, and embed a watermark information bit if its difference is less than a certain threshold. However, they have a trouble to embed a watermark in a noisy region because most of the differences in such a region are large. Such trouble is also occurred at the embedding in prediction errors, but it may not so serious compared with the conventional ones because the predictor may be able to output relatively small prediction errors for noisy regions. In order to apply the basic embedding operation for



the prediction errors, several parameters must be modified carefully. Thodi and Rodriguez[9] applied a predicted value, but the predictor is not a well-designed one. Although their results show that the capacity is large, their method is not reversible because a location map is not considered.

### 2.3 Definitions

On the expansion of the differences in the conventional schemes, overflow and underflow of pixel values are avoided by carefully selecting the target pair/vector of pixels. The selection is based on the definitions of *expandable* and *changeable*, and it is performed in the transformed domain. Since the definitions and classifications are essentially same as that of our scheme, we describe the detail of our definitions. The main concern of our scheme is the embedding capacity for each pixel/vector. Tian's algorithm is capable of embedding as high as 1/2 bits/pixel because one bit is embedded in the difference of pair of pixels, and Alattar's one is as high as  $(n - 1)/n$  bits/pixel for each difference of  $n$  pixels. On the other hand, our scheme can be at most 1 bits/pixel as pixel-wise operation is possible by applying the prediction error instead of differences among pixels.

It seems difficult to modify the prediction error using the previously occurred pixels  $a, b, c$ , and  $d$  because those pixels are also used for the prediction of other pixels. Instead, we modify the current target pixel  $x$ , which is easily calculated from Eq.(1).

$$x_e = x + \epsilon + w \quad (3)$$

Here, it must be considered that the pixel value must be in  $[0, 255]$ , otherwise it causes overflow(more than 255) or underflow(less than 0). In order to control the embedding operation, the following definition is introduced.

**Definition 1.** *The pixel  $x$  is said to be expandable if, for any  $w \in \{0, 1\}$ ,  $x$  can be modified to  $x_e$  without causing overflow and underflow.*

If a pixel is expandable, it is possible to embed an information bit in the LSB by expanding the prediction error  $\epsilon$ . However, when one wants to extract an information, it is impossible to find if the pixel was expandable or not before the embedding. By considering the extraction, the prediction errors of the pixels which are not expandable are modified to even number by the following equation.

$$x^* = \begin{cases} x - 1 & \text{if } \epsilon \text{ is odd} \\ x & \text{otherwise.} \end{cases} \quad (4)$$

Then the LSB of the prediction error is removed, and the information is embedded with a watermark information in an image so as to be reversible. Here, the overflow and underflow of  $x^*$  must be considered. Therefore, the following definition is also introduced.

**Definition 2.** *The pixel  $x$  is said to be changeable if, for any  $w \in \{0, 1\}$ ,  $x$  can be modified to  $x_c$  without causing overflow and underflow.*

$$x_c = x^* + w \quad (5)$$

Notice that all expandable pixels are changeable and they are still changeable after the embedding. Based on the characteristic, for both expandable and changeable pixels, information bits are embedded into the LSBs of their prediction errors, which can be extracted from all changeable pixels of the watermarked image. Although the watermark can be extracted, it is impossible to determine if each pixel was expandable or changeable before embedding. Hence, the information about the original conditions of pixels which is called location map is embedded in addition to a watermark. Here, a lossless compression algorithm, such as JBIG2 and an arithmetic compression algorithm, is performed to reduce the size of the map. As the consequence, the embedding information bits are composed of three parts; a compressed location map, LSB of  $\epsilon$  of changeable pixel, and a watermark information.

In general, there is a trade-off between the distortions and the capacity in watermarking technique, and it is desirable that the trade-off is controlled for applications. It is achieved by introducing a threshold  $T$  for the determination of expandable or not. If the absolute value of a prediction error is less than  $T$  and Definition 1 is satisfied, the pixel is regarded as expandable. Since the changes caused by the operation at expandable pixels are restricted less than  $T$ , the degradation of the quality is controlled.

Each pixel can be classified into three groups according to the Definition 1 and 2. The first group  $S_1$  contains all expandable pixels whose prediction errors less than a predefined threshold  $T$ . The second group  $S_2$  contains all changeable pixels that are not in  $S_1$ . The third group  $S_3$  contains the rest of the pixels which implies not changeable. Also, let  $S_4$  denote all changeable pixels ( $S_4 = S_1 \cup S_2$ ).

### 3 Proposed Reversible Watermarking Algorithm

In this section, we propose a new reversible watermark scheme using the predictive coding technique in JPEG-LS. A basic algorithm of the reversible operation for the embedding is shown.

#### 3.1 Embedding a Reversible Watermark

The proposed algorithm is composed of two parts for the embedding of a watermark, one is *formatting*, and the other is *embedding*. The summary of the operation is shown below.

**Formatting:** For a scanned pixel  $x_{i,j}$ , ( $0 \leq i \leq N - 1, 0 \leq j \leq M - 1$ ) by a raster scan order, the following operations are performed.

1. Calculate the prediction error  $\epsilon_{i,j}$ .
2. Modify  $x_{i,j}$  to  $\bar{x}_{i,j}$  based on the following three conditions.

$$\bar{x}_{i,j} = \begin{cases} x_{i,j} + \epsilon_{i,j} & \text{if } x_{i,j} \in S_1 \\ x_{i,j} - 1 & \text{if } x_{i,j} \in S_2 \text{ and } \epsilon_{i,j} \text{ is odd} \\ x_{i,j} & \text{otherwise} \end{cases} \quad (6)$$

The modification implies,

$$\bar{\epsilon}_{i,j} = \begin{cases} 2\epsilon_{i,j} & \text{if } x_{i,j} \in S_1 \\ \epsilon_{i,j} - 1 & \text{if } x_{i,j} \in S_2 \text{ and } \epsilon_{i,j} \text{ is odd} \\ \epsilon_{i,j} & \text{otherwise.} \end{cases} \quad (7)$$

After the above modification, the prediction error  $\bar{\epsilon}_{i,j}$  becomes even number if  $x_{i,j} \in S_4$ .

3. The location map is set to  $L_{i,j} = 0$  if  $x_{i,j} \in S_1$ , otherwise  $L_{i,j} = 1$ .
4. If  $x_{i,j} \in S_2$ , then the LSB of  $\epsilon_{i,j}$  is added to a vector  $\mathbf{B}$  as one element.

Note that for the prediction of a current pixel  $x_{i,j}$ , the previously formatted four pixels which are specified in Fig.1 are used.

The bit-stream of the location map  $L_{i,j}$  is compressed by JBIG2. We call the bit-stream of the compressed map  $\mathbf{L}$ . Then,  $\mathbf{L}$ ,  $\mathbf{B}$ , and a watermark information bit-stream  $\mathbf{W}$  are embedded. Combining those bit-streams, the embedding information  $\mathbf{w}$  is produced.

$$\mathbf{w} = \{\mathbf{head} \parallel \mathbf{L} \parallel \mathbf{B} \parallel \mathbf{W}\} \quad (8)$$

$$= \{w_t | 1 \leq t \leq \sigma\} \quad (9)$$

Where **head** is a header file of the embedding information,  $\parallel$  means concatenation, and  $\sigma$  is the bit-length of  $\mathbf{w}$ . Here,  $\sigma$  can be represented as

$$\sigma = \text{len}(\mathbf{head}) + \text{len}(\mathbf{B}) + \text{len}(\mathbf{L}) + \text{len}(\mathbf{W}), \quad (10)$$

$$= N_1 + N_2, \quad (11)$$

where the function  $\text{len}(x)$  outputs the bit-length of  $x$ , and  $N_1$  and  $N_2$  are the number of pixels in  $S_1$  and  $S_2$  respectively.

**Embedding:** After the above formatting operation, every prediction error, not pixel value, becomes even number. The embedding operation is simply to add  $w_t$  directly to each pixel  $x_{i,j} \in S_4$  by a raster scan order, which implies to insert  $w_t$  into the LSB of the formatted prediction error  $\bar{\epsilon}_{i,j}$ .

By setting a counter  $t = 1$ , the following operations are performed repeatedly for each formatted pixel  $\bar{x}_{i,j}$ .

1. Modify  $\bar{x}_{i,j}$  to  $x'_{i,j}$  using the embedding information bit  $w_t$

$$x'_{i,j} = \begin{cases} \bar{x}_{i,j} + w_t & \text{if } x_{i,j} \in S_4 \\ \bar{x}_{i,j} & \text{otherwise} \end{cases} \quad (12)$$

2. Increment  $t = t + 1$  if  $x_{i,j} \in S_4$ .
3. If  $t \leq \sigma$ , go back to the step 1, otherwise quit.

### 3.2 Extraction and Recovery

On a reversible watermark, an original image is recovered from a watermarked image using an embedded information. Therefore, a watermark is first extracted from a watermarked image, and then the original image is recovered.

**Extraction:** On the prediction of JPEG-LS, the scanning order is very important to recover the original image, and it is performed by a raster scan order. When the embedded information is extracted, each information bit is extracted by this order. Since each prediction error is calculated from the previously formatted four pixels, the same pixels are required for the prediction at the extraction. Therefore, the extraction is performed by the following steps for each raster scanned pixel.

1. Set a counter  $t = 1$ .
2. Calculate the prediction error  $\epsilon_{i,j}$  for a target pixel  $x'_{i,j}$ .
3. If  $x'_{i,j}$  is changeable, which implies  $x_{i,j} \in S_4$ , then the following operations are performed.
  - 3-1. Extract the LSB of  $\epsilon'_{i,j}$  as  $t$ -th embedding information bit  $w_t$ .

$$w_t = \epsilon'_{i,j} \pmod{2}, \quad (13)$$

- 3-2. In order to make the prediction error even number, subtract  $w_t$  from  $x'_{i,j}$ .

$$\bar{x}_{i,j} = x'_{i,j} - w_t \quad (14)$$

- 3-3. Store the re-formatted prediction error  $\bar{\epsilon}_{i,j}$ .

$$\bar{\epsilon}_{i,j} = \epsilon'_{i,j} - w_t \quad (15)$$

- 3-4. Increment  $t = t + 1$ .
4. Perform the above step 2 and step 3 using the re-formatted four pixels until all pixels are checked.

**Recovery:** If the embedding information  $\mathbf{w}$  is completely extracted and the original formatted image is recovered, then the original image is recovered using  $\mathbf{w}$  and each re-formatted prediction error  $\bar{\epsilon}_{i,j}$ . The procedure is described below.

1.  $\mathbf{w}$  is divided into three bit-streams,  $\mathbf{L}$ ,  $\mathbf{B}$ , and  $\mathbf{W}$  using the header file *head* which is predefined bits from the top of  $\mathbf{w}$ .
2. Stretch  $\mathbf{L}$  to obtain a location map  $L_{i,j}$ .
3. Using  $\mathbf{B} = \{B_t | 1 \leq t \leq \text{len}(\mathbf{B})\}$ , each original pixel  $x_{i,j}$  is recovered.

$$x_{i,j} = \begin{cases} \bar{x}_{i,j} - \frac{\bar{\epsilon}_{i,j}}{2} & \text{if } L_{i,j} = 0 \\ \bar{x}_{i,j} + B_t & \text{if } L_{i,j} = 1 \text{ and } x'_{i,j} \text{ is changeable} \end{cases} \quad (16)$$

### 3.3 Capacity

The capacity of our scheme is dependent on the number of pixels in  $S_1$  and the compression ratio of the location map. In our scheme, each embedding information bit  $w_t$  is inserted into the LSB of the prediction error of the corresponding pixel in  $S_4$ . Notice that for a pixel in  $S_2$ , the LSB is just replaced by  $w_t$  and the LSB is preserved in  $\mathbf{B}$ , which can not be compressed in theory because of its randomness.  $\mathbf{B}$  is, therefore, directly embedded and the bit-length  $\text{len}(\mathbf{B})$  must be equal to  $N_2$ . As the result, the bit-length of the watermark information is represented as follows using Eq.(10) and Eq.(11).

$$\text{len}(\mathbf{W}) = N_1 - \text{len}(\text{head}) - \text{len}(\mathbf{L}). \quad (17)$$

## 4 Enhancement of the Performance

For the improvement of the capacity, one simple method is to increase  $N_1$  in Eq.(17) by enlarging a threshold  $T$ , but it causes the degradation of an image. In order to increase the capacity without degrading the perceptual quality, we propose two methods to reduce the size of compressed location map,  $len(\mathbf{L})$ , in this section.

### 4.1 Vectorization

Since the embedding operation is performed for each pixel in the basic scheme, each pixel needs the corresponding location map, which becomes the same size of an image. Although it is compressed, the size may be still large. In order to decrease the location map itself, several pixels are put together into one vector which is judged expandable if all pixels in the vector are in  $S_1$ . Such pixels should be selected carefully from an image in our scheme, because the raster scan order must be followed. Generally, there is a strong mutual relation among neighboring pixels, the vectorization can increase the capacity efficiently.

1. For successive  $m$  pixels, the formatting operation is performed.
2. If all  $m$  pixels are in  $S_1$ , a reduced location map is set to  $\ell_{i,j/m} = 0$  and go to the next  $m$  pixels. Otherwise,  $\ell_{i,j/m} = 1$  and performs the step 3 to step 5 using the original  $m$  pixels successively.
3. Calculate the prediction error  $\epsilon_{i,j}$  based on Eq.(1).
4. Modify  $x_{i,j}$  to  $\bar{x}_{i,j}$  based on the following two conditions.

$$\bar{x}_{i,j} = \begin{cases} x_{i,j} - 1 & \text{if } x_{i,j} \in S_4 \text{ and } \epsilon_{i,j} \text{ is odd} \\ x_{i,j} & \text{otherwise} \end{cases} \quad (18)$$

5. If  $x_{i,j} \in S_4$ , then the LSB of  $\epsilon_{i,j}$  is added to a vector  $\mathbf{B}$  as one element.

After the above operations, the produced location map consists of  $N \times M/m$  elements. Therefore, the size of the location map becomes  $1/m$  compared with the basic one. The vectorization method reduces the size of location map with a little sacrifice of the capacity.

In the vectorized scheme, each vector is classified according to the Definition 1, Definition 2, and a threshold  $T$ . The threshold controls the trade-off between the capacity of a watermark information and the distortions caused by the embedding. If at least one pixel in a vector is not in  $S_1$ , the vector is regarded as non-expandable. Then, instead of expansion, each pixel in the vector is modified as a changeable or non-changeable ones. It is remarkable that the decoder of the vectorized scheme can apply the same one as the basic scheme because the embedded information is extracted from the LSB of the prediction errors of pixels in  $S_4$ . In order to recover the original image, the applied method must be informed, which is easily realized by adding such information to the header file *head*.

## 4.2 Composition of Location Map

In order to reduce the size of  $\mathbf{L}$  more effectively, we reconsider the composition of the map. In the basic scheme, the map is merely produced by putting “0” or “1” symbol to each element if a target pixel is in  $S_1$  or not, and the information is required for the recovery for the original pixel because the operation is dependent on the pixel if it was in  $S_1$  or not. Here, it is remarkable that several pixels are still in  $S_1$  after the embedding. For such pixels, the location map is not required for the recovery operation because they are determined by themselves.

When a pixel  $x$  is in  $S_1$  and its prediction error  $\epsilon$  satisfies an inequality  $|\epsilon| < T/2$ , the formatted pixel  $\bar{x}$  also belongs to a group  $S_1$  for the formatted prediction error  $\bar{\epsilon}(= 2\epsilon)$  if  $x_{ae}$ ,

$$x_{ae} = \bar{x} + 2\epsilon + w, \quad (19)$$

$$= x + 3\epsilon + w, \quad (20)$$

does not cause overflow and underflow, where  $w \in \{0, 1\}$ . In order to classify such pixels, we define *absolutely expandable*.

**Definition 3.** *The pixel  $x$  is said to be absolutely expandable if, for any  $w \in \{0, 1\}$ ,  $x$  can be modified to  $x_{ae}$  without causing overflow and underflow, and  $|\epsilon| < T/2$  is satisfied.*

For a pixel  $x' \in S_2$ , there are two possible candidates for the group to which the original pixel  $x$  belonged, namely  $S_1$  and  $S_2$ . There are also several pixels in  $S_2$  that the original group can be determined by themselves if those prediction error satisfy an inequality  $|\epsilon| > 2T$  because of the following reason. When a pixel  $x$  belongs to a group  $S_1$ , the prediction error  $\epsilon$  is less than  $T$ . It means that after the formatting operation, the modified prediction error  $\bar{\epsilon}(= 2\epsilon)$  must be less than  $2T$ . Therefore, if  $|\epsilon| > 2T$ , such a pixel must be in  $S_2$ , which is also hold in the recovery operation. For the classification, we define *absolutely changeable* as follows.

**Definition 4.** *The pixel  $x$  is said to be absolutely changeable if  $x$  is changeable and  $|\epsilon| > 2T$  is satisfied.*

By introducing the Definition 3 and 4, several pixels in  $S_1$  and  $S_2$  can be belongs to an extra group  $S_5$  which contains absolutely expandable pixels and absolutely changeable pixels. Although each pixel  $x_{i,j}$  has a corresponding location map  $L_{i,j}$  in the basic scheme, the map in pixels in  $S_5$  can be omitted, which contributes on the reduction of the size of  $len(\mathbf{B})$ . One simple method for the reduction is merely to remove the corresponding information of the map. Considering the compression algorithm of JBIG2, however, the map should be two-dimensional and hence it causes problem. Instead, we manage to compose the location map in order to enhance the performance of JBIG2.

First, we compose the location map by the following rules.

$$L_{i,j} = \begin{cases} * & \text{if } x_{i,j} \in S_5 \\ 0 & \text{if } x_{i,j} \in S_1 \text{ except } x_{i,j} \in S_5 \\ 1 & \text{otherwise} \end{cases} \quad (21)$$

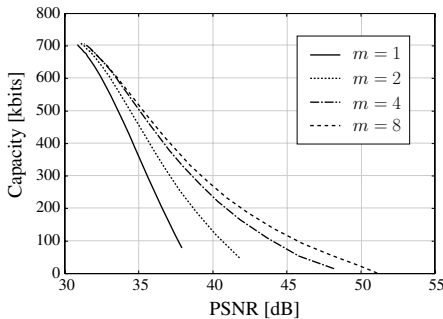
Where the symbol “\*” indicates “0” or “1” dependent on the contexts. It is certain that higher compression ratio can be achieved by adaptively setting the symbol “\*”. In this paper, for the adaptive modification of the map, the following method is applied. We merely set all the symbols to “0” or “1”, compress the two types of the modified map, and adopt the better one. Such selection is very simple, but it contributes greatly for the reduction of the compressed data size.

If the location map is composed by the above procedure, each pixel  $\bar{x}_{i,j}$  must be checked whether  $x_{i,j}$  is in  $S_5$  before a recovery operation is performed. If  $\bar{x}_{i,j}$  is expandable for its prediction error  $\bar{\epsilon}_{i,j}$  which is less than  $T$ ,  $x_{i,j}$  must be absolutely expandable. And if  $|\bar{\epsilon}_{i,j}|$  is more than  $T$ ,  $x_{i,j}$  must be absolutely changeable. For such cases, the recovered location map from the extracted  $w$  is not referred for the recovery of the pixels. Notice that a threshold  $T$  is necessary to judge a pixel absolutely expandable/changeable. Therefore, such information should be added to the header file *head*.

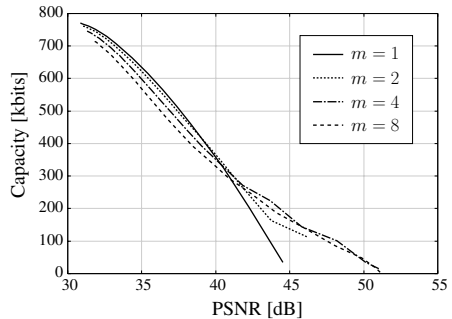
### 5 Experimental Results

We have implemented our algorithm and estimated the capacity and distortions with the basic scheme and the enhanced scheme. The images used in the evaluation are “lena”, “baboon”, “fruits”, and “F16” with RGB color of  $512 \times 512$  pixels. We tested the algorithm for each RGB color components respectively with the same threshold  $T$ . In the following simulation, the capacity is calculated by omitting the size of *head* as it is negligibly small (It may be less than 100 bits).

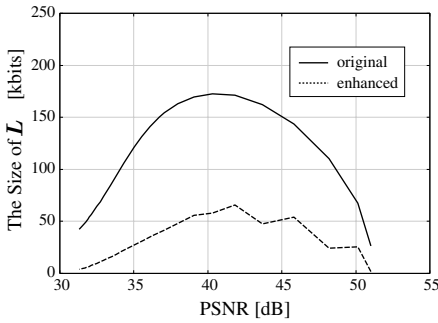
The capacity obtained with various vector size  $m$  for the image “lena” is plotted against the PSNR in Fig.3. This figure reveals that the capacity is increased according to the increase of the vector size  $m$ . The similar results are obtained from other images. To achieve a large capacity, the vectorization seems one method for the improvement of the basic scheme. Next, the capacity of the enhanced scheme which modifies the composition of the location map is shown in Fig.4. It is clear



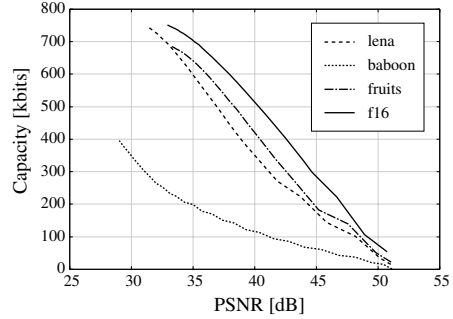
**Fig. 3.** The comparison of the performance for the number of element of the composed vector for an image “lena”



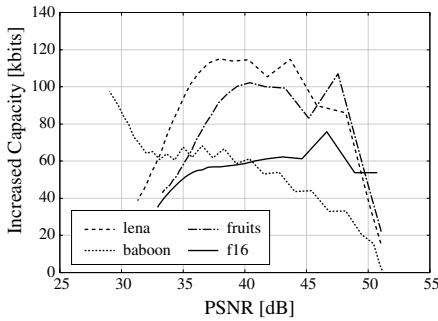
**Fig. 4.** The capacity improved by the modification of a location map for an image “lena”



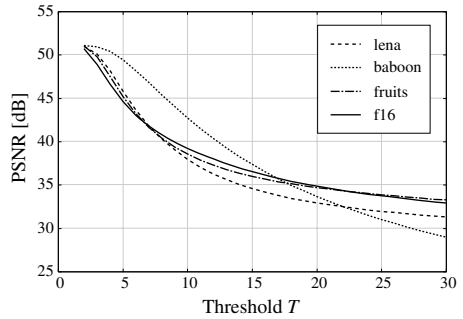
**Fig. 5.** The comparison with the compressed data size of a location map for an image “lena”



**Fig. 6.** The capacity of embedding information versus PSNR



**Fig. 7.** The improved capacity by reducing the size of the location map.

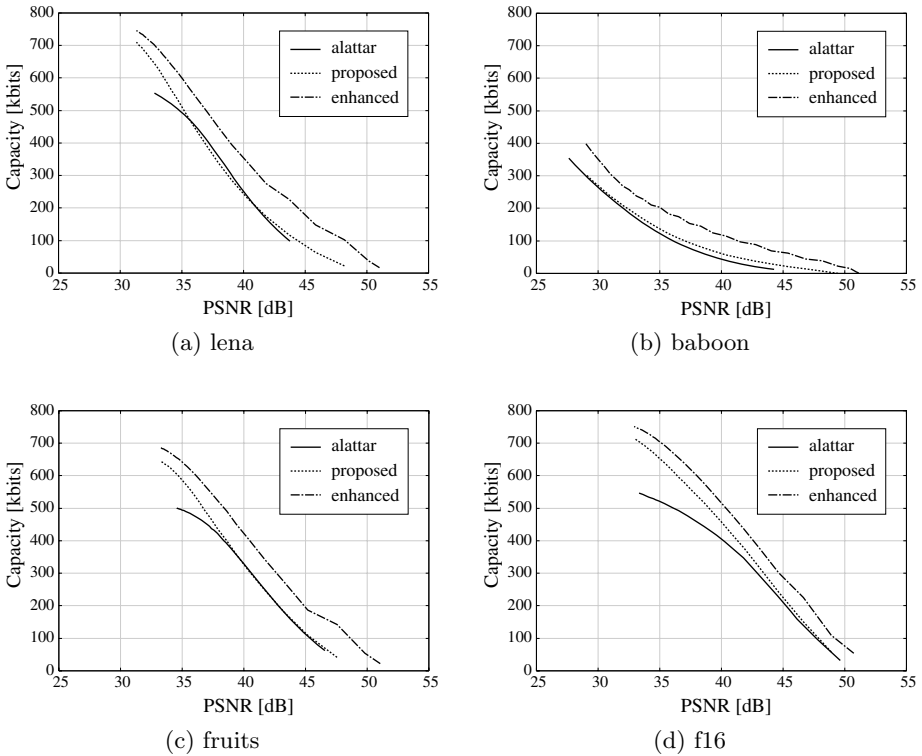


**Fig. 8.** The relation between PSNR and threshold  $T$ .

that the capacity is improved at all range compared with the basic scheme and its vectorized scheme. It is remarkable that the largest capacity can be achieved for a basic non-vectorized scheme even if it can not make a space to embed a watermark when the value of PSNR is high. If one wants to get better performance for overall range, we recommend to use  $m = 4$ . In the rest of this section, the results are obtained for the constant parameter  $m = 4$ .

The effects of the compression in the modified location map is numerically estimated. Figure 5 shows the size of the compressed location map for both the basic scheme and the enhanced scheme. From the result, the location map is well compressed by JBIG2 when the the map is adaptively composed. Since our method simply assigns “0” or “1” symbol to the location map based on the new conditions *absolutely expandable* and *absolutely changeable*, a further optimization can be achieved by composing the map more adaptively considering the applied compression algorithm, which is our future work. For the evaluation of other images, the capacity of the enhanced scheme is shown in Fig.6. The results





**Fig. 9.** The comparison of the capacity, where the vector is composed from 4 pixels ( $m = 4$ )

reveal that the capacity is variable for images. Although the performance of the predictor in JPEG-LS at a flat region is superior to noisy region in a image, a watermark is spread all over the image. The effects caused by the modification of the location map is numerically estimated, and the amount of increased capacity is shown in Fig.7. The contribution in the increase of the capacity is dynamically changed for each image and each PSNR because of the characteristic of the compression algorithm of JBIG2.

When a watermark is embedded, a kind of sharpening effects is appeared and the effects grow stronger for the increase of the amount of watermark information. For the numerical evaluation of the image quality, the relation between PSNR and threshold  $T$  is shown in Fig.8. The size of threshold implies the amount of maximum changes caused by the embedding. As the images “lena”, “fruits”, and “f16” contains a lot of flat regions, the prediction errors are distributed in a small range, and hence the curve is slowly decreased and reached its lower band for rather small  $T$ . On the other hand, as most parts of the image “baboon” are noisy, the curve becomes rapidly down.

We also compare the performance of our scheme with that of Alattar[5] which achieves a large capacity at the state-of-art schemes. In the work, largest capacity

can be obtained when three bits are embedded in each vector of four pixels, which is similar to our scheme as a better performance is obtained for  $m = 4$ . Under such conditions, the comparisons are shown in Fig. 9. The results clarify that our enhanced scheme is superior to the conventional one for all test images.

## 6 Conclusion

In this paper, a reversible watermark with a very large capacity based on the prediction errors calculated in JPEG-LS has been proposed. Since the prediction errors follow TSGD model centered at zero, the distortions caused by our embedding is kept small. In addition, a watermark is spread all over the image in our scheme, though the conventional schemes embed mainly in the flat region. Those properties contribute to the improvement of the perceptual quality. In order to improve the capacity, we compose a vector from several pixels and the size of location map is decreased effectively exploiting the characteristic of a pixel/vector. From our simulation results, a better performance can be obtained when successive four pixels are treated as a vector. Our future work is to produce a location map adaptively considering the applied compression algorithm, and to try other predictive coding for our technique.

## References

1. S. Katzenbeisser and F. A. P. Petitcolas, *Information hiding techniques for steganography and digital watermarking*. Artech house publishers, Jan. 2000.
2. M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-LSB data embedding," *IEEE Trans. Image Process.*, vol.14, no.2, pp.253-266, 2005.
3. J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol.13, no.9, pp.890-896, 2003.
4. H. Heijmans and L. Kamstra, "Reversible data embedding based on the haar wavelet decomposition," *Proc. of DICTA2003*, pp.5-14, 2005.
5. A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. Image Process.*, vol.13, no.8, pp.1147-1156, 2004.
6. M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol.9, no.8, pp.1309-1324, 2000.
7. A. Netravali and J. O. Limb, "Picture coding: a review," *Proc. of IEEE*, vol.68, pp.366-406, 1980.
8. S. W. Golomb, "Run-length encodings," *IEEE Trans. Inform. Theory*, vol.IT-12, pp.399-401, 1966.
9. D. M. Thodi and J. J. Rodriguez, "Reversible watermark by prediction-error expansion," *Proc. of 6th IEEE Southwest Symposium on Image Analysis and Interpretation*, pp.21-25, 2004.

# PCAV: Internet Attack Visualization on Parallel Coordinates\*

Hyunsang Choi and Heejo Lee \*\*

Korea University, Seoul 136-713, South Korea  
{realchs, heejo}@korea.ac.kr

**Abstract.** This paper presents PCAV (Parallel Coordinates Attack Visualizer), a real-time visualization system for detecting large-scale Internet attacks including Internet worms, DDoS attacks and network scanning activities. PCAV displays network traffic on the plane of parallel coordinates using the source IP address, destination IP address, destination port and the average packet length in a flow. These four values are used to draw each flow as a connected line on the plane and surprisingly a group of lines forms a particular shape in case of attack. Thus, a simple but novel way of displaying traffic reveals ongoing attacks. From the fact that numerous types of attacks form a specific pattern of graphs, we have developed nine signatures and their detection mechanism using an efficient hashing algorithm. Using the graphical signatures, PCAV can quickly detect new attacks and enables network administrators to instantly recognize and respond to the attacks. Another strength of PCAV comes from handling flows instead of packets. Per-flow visualization greatly reduces the processing time and further provides compatibility with legacy routers which export flow information such as Net-Flow in Cisco routers. We have demonstrated the effectiveness of PCAV using real attack traffics.

## 1 Introduction

Explosive expansion of computer networks has the benefit of providing much improved accessibility to a wide array of valuable data. However, the number of incidents is increasing over time. Many intrusion detection technologies have been proposed, but still have some inherent weaknesses. Conventional intrusion detection systems, which are based on known attack signatures, cannot detect unknown attacks. Further, intrusion detection systems based on anomaly detection mechanisms often generate a huge number of false alarms which overwhelm security engineers. Moreover, conventional monitoring systems such as IDS's and firewalls, provide a rudimentary level of displaying results visually.

One promising approach is visualization to handle complex situations, using a simple and intuitive method [4]. Several approaches to information visualization are studied widely [2], due to the well-known advantages resulting from visualization. Visual

---

\* This work was supported in part by the ITRC program of the Korea Ministry of Information & Communications.

\*\* To whom all correspondence should be addressed.

images can be obtained from raw data using computer graphics techniques and algorithms. From these images, valuable insights can be acquired. It is the efficient link from the human mind to the modern computer, which represents key technology for extracting information. This visual representation is becoming more and more essential in the field of network intrusion detection [10].

We introduce a simple but novel way for visualizing Internet attacks on parallel coordinates. Parallel coordinates have many good properties such as representing more than three fields in a two dimensional space [1]. In order to visualize most notorious attacks such as Internet worms, we have carefully selected four important fields available on each flow. As a result, we can develop nine graphical signatures to detect ongoing attacks, which include Internet worms, DDoS attacks and network scanning attacks. As well, we devise an  $O(1)$  hashing algorithm to identify these signatures. The effectiveness of the proposed visualization approach is shown by running on real network traffic and revealing hidden attacks as a visual way.

We use flows for input data, instead of packets, because of system performance and compatibility with legacy routers. A flow is a single network connection and can consist of millions of packets. Handling flow-level information greatly reduces the processing time so that enables to run on high-speed links. Furthermore, many legacy routers provide flow information and they are widely deployed, which includes Net-Flow in Cisco routers. This compatibility with legacy routers greatly enhances the usability of the visualization mechanism.

The aim of this study is not to propose a new visualization technique. The main contribution is how to use parallel coordinates to display Internet attacks. Displaying network flows using carefully chosen values forms a unique graphical image for each attack. It is shown that this mechanism works for detecting notorious Internet attacks such as rapidly spreading Internet worms.

The following section shows the benefits of visualization approaches. The characteristics of Internet attacks are discussed, in particular, what data should be visualized and how this data should be visualized to make a unique shape. Section 3 describes some patterns of visualized graphs and signatures from the patterns. Then, we discuss several data structures and propose a hash algorithm to identify attack signatures. The evaluation of the mechanism is done in Section 4.

## 2 Attack Visualization

Humans, as a visual being, can easily recognize and infer patterns from visual aids intuitively. This section describes the benefits of attack visualization, and the principles of our visualization approaches used to display Internet attacks.

### 2.1 Benefits of Attack Visualization

There are four main benefits when applying information visualization to the problem of intrusion detection. First, attack visualization can easily deal with highly heterogeneous and noisy data. Network traffic is extremely complex and must be correlated with several variables such as source address, destination address, port number, packet length,

and TCP flag, but attack visualization enables us to present the traffic situation in an intuitive way. Second, attack visualization requires no understanding of complex mathematical or statistical algorithms. Visual images give perceptual clues faced with an attack. Third, attack visualization allows us to gain valuable insight into the analyzed data and deduce new hypotheses. Therefore, even though an unknown attack may have occurred, if an image pattern (signature) from the unknown attack is obtained, the attack can be quickly detected. Finally, attack visualization can be much faster than other anomaly detection approaches. Many anomaly detections require training and comparing with history, but attack visualization can quickly identify an attack using pre-defined image patterns.

## 2.2 Attack Characteristics

In order to devise a visual mechanism for most popular Internet attacks such as DDoS attacks, worm attacks, or network scans, their characteristics must be considered in terms of visualization. Fortunately, these notorious attacks have one common characteristic, which is called "one-to-many relationship" between attackers and victims. While legitimate flows have one-to-one relationship, attack flows have a one-to-many relationship. This is a good point for visualizing the attacks.

A DoS attack is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the targeted network or overloading the computational resources of the targeted system. In a distributed DoS attack, the attacking hosts are often personal computers with broadband connections to the Internet that have been compromised by viruses or Trojan horse programs, allowing the perpetrator to remotely control the machine and direct the attack, often through a botnet. With enough slave hosts, the services of even the largest and most well connected website can be denied. Therefore, in a DDoS attack, there are many attackers and one victim, which forms a one-to-many relationship between a victim (destination) and the attackers (source).

A worm is defined as self-propagating malicious code. Once a machine is infected, target hosts are acquired by pseudo random number generators, or host scans to detect vulnerable machines (usually with a single vulnerability) in a certain network. If targets are decided at a previous step, an infected machine propagates worm code to targets. Therefore, a worm represents a one-to-many relationship between the infected machine (source) and the victims (destination).

Network scanning, used by hackers to probe hosts, also exhibits steps in worm propagation, and has a one-to-many relationship between a hacker (source) and scanned network hosts (destination). Port scanning is a method used for probing available services of a certain host. There is one attacker, one or many hosts, and many scanned ports. Therefore, a port scan is also characterized as a one-to-many relationship.

## 2.3 Four Fields as Attack Parameters

In the previous subsection, an important characteristic was presented, namely, one-to-many relationship. Now, we should consider which manifested variables display these important characteristics of attacks.

First, an attack may consist of attacker(s) and victim(s), therefore, we select the source IP address and destination IP address in a flow information. These two values can be used to visualize the particular characteristic of attacks. These values are also stored in the fields of every packet header so that they can be used to distinguish the attacking packets from legitimate packets.

Second, an attack usually targets one or more ports in TCP or UDP protocols so that the destination port number is selected as a parameter. This value identifies the targeted service of an attack and verifies port scanning attacks. On the contrary, the source port number is less meaningful than the destination port number since the source port is chosen randomly among available ports.

Third, the average size of packets in a flow can be used as a parameter. This information gives some clues whether the flow is suspicious or not. Network scanning and DDoS attacks exploit a flooding procedure and the procedure normally uses empty packets without payload. These attacking packets usually have a packet length of 40 or 48 bytes. Contrarily, Internet worms have a payload to exploit the vulnerabilities they can use. These worms are characterized by a fixed length of code because the worm codes on the payload are unique for each worm. Polymorphic worms may change their code, as an attempt to evade signature-based systems, but until now, every active worm has characterized by a unique payload.

Finally, we picked the TCP flag as an additional parameter. Network scanning and DDoS attacks may send one packet repeatedly so that its TCP flag has the same value. Thus, we can classify attack traffic or normal traffic using its TCP flag. For instance, some normal traffic have a one-to-many relationship, such as P2P communications, and they can be considered as legitimate traffic after comparing their TCP flag with that of a normal TCP handshaked flow.

## 2.4 Per-Flow Visualization

A visualization system can use a flow, instead of a packet, as a basic unit of data because it drastically reduces processing time without loss of necessary information. Furthermore, per-flow visualization provides the compatibility with legacy routers so that we can deploy the system without the change of current networks. One good example is to run a visualization system with Cisco routers which exports NetFlow information [5]. This implies the visualization system can use Cisco routers as sensors.

A flow can be defined as a set of packets with the same source IP, destination IP, source port and destination port that can be thought of as a connection between two computers. Available information on each flow includes the above mentioned attack parameters which are source address, destination address, destination port, average packet length, and the cumulative OR of TCP flags.

## 2.5 Parallel Coordinates

One important aspect of information visualization is scalability. Parallel coordinates provide great scalability to multiple dimensions. Parallel coordinates are not complex and allow multi-parameter patterns to be analyzed. We use parallel coordinates and a scattered plot matrix on the coordinates as the initial plane for visualizing Internet

attacks. This combined method also results in a quicker understanding and a more informational graph over that of a scattered plot matrix.

This visualization technique has many advantages and they are listed as follows. First, this technique does not give preference to any specific dimension. An important feature, if there is no evidence regarding which dimension is more important, is that by default there is no bias towards any specific dimension. Second, both methods have no limit to the number of parameters that can be visualized. Therefore, we can deal with a number of variables that we desire to visualize and can easily add new visualization parameters. Third, both mechanisms prominently show trends, correlations and divergences from the raw data. Therefore, this advantage enables us to gain critical insight into the flow of data and present reliable intuitive hypotheses. Using this method, even if an unknown attack occurs, a specific image pattern is obtained from the unknown attack and the attack can be detected in a timely manner. Fourth, both techniques can handle even continuous and categorical data (though some of the important benefits may be lost).

### 3 Parallel Coordinates Attack Visualization (PCAV)

#### 3.1 Attack Signatures

Here we show how parallel coordinates can be used to describe an attack in a more informational graph pattern. The coordinates represent four different parameters in a flow. The first represents the source address of each flow, the second represents the destination address, the third represents the destination port and the fourth represents the average packet length of each flow. These four values shown in each flow, enables the flow to be shown as a connected line with parallel coordinates. Therefore, one connected line represents one connection. As shown in Section 2.2, the attack characteristics, of each attack have a one-to-many or many-to-one relationship with each parameter.

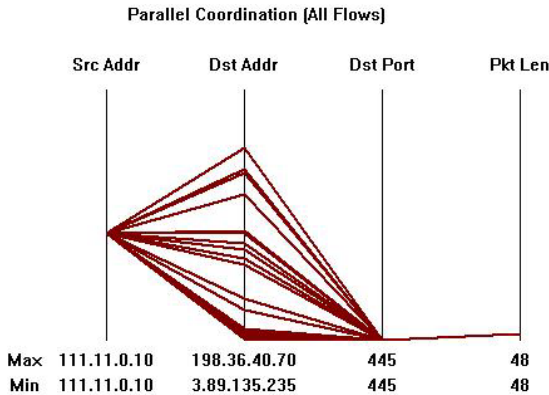











Fig. 1. The attack graph of a host scan

In order to know the graph pattern for each attack, let us consider a host scanning attack. In a host scan, an attacker may want to know which hosts are alive in the target network. The sequence of a host scan progresses gradually to check the targeted destination port of each host. Every packet usually has no payload, for more effective scans. Thus, the sizes of scanning packets can be 40 or 48 bytes only for TCP and IP headers. Occasionally scanning programs also use the TCP option of selective acknowledgment (SACK), which is commonly used to allow senders' TCP to employ more advanced loss recovery and congestion control. In these cases, the sizes of scanning packets become 48 bytes. The graph of host scans looks like a fish (diamond-line pattern) as shown in Fig. 1.

**Table 1.** Attack signatures of nine attacks

Implied Attack	Signature	Divergences
Portscan		1:1:m:1
Hostscan		1:m:1:1
Worm		1:m:1:1
Source-spoofed DoS (port fixed)		m:1:1:1
Backscatter		1:m:m:1
Source-spoofed DoS (port varied)		m:1:m:1
Distributed hostscan		m:m:1:1
Network-directed DoS		m:m:m:1
Single-source-spoofed DoS		1:1:1:1

Not only host scans but also other attacks in one-to-many relationships show an interesting graph pattern, which are shown in Table 1. This table describes the attack signature of each implied attack and its divergences (patterns of one-to-many and many-to-one relationships). For example, in a port scan, there is one attacker and one victim, and the attacker wants to know which ports are open. To accomplish this, the attacker may use a port scanning program which checks the destination port of the victim one by one, sequentially or randomly. This behavior represents 1:1:many:1 patterns and the signature graph pattern looks similar to a kite (line-diamond) as shown in Table 1.

Average packet lengths can be used to distinguish similar attack patterns. A worm and a host scan have the same graph patterns (diamond-line). But a host scan may have no payload, whereas a worm should have a payload to infect other machines. Thus, the average packet lengths of all flows in a worm epidemic are constant and relatively larger than the average packet length of a flow in a host scan, which is 48 bytes in Fig. 1.



Backscatter is not actually an attack, but a reflective state. This state can be used to detect attacks, so it is added to the signature table. The source spoofed DoS (port fixed) is a DDoS attack using a fixed port. Usually the attack has no payload so the graph pattern is represented as a triangle with a connected line. However, a source spoofed DoS (port varied) is a DDoS attack using randomly chosen destination ports and as usual, has no payload. Therefore, the represented graph looks like a rightward looking fish. A distributed host scan is multiple host scanning behavior. Network-directed DoS is a DDoS attack targeted at a specific network, so destination addresses (victims) are network scale.

### 3.2 System Design

In order to display and detect ongoing attacks using the attack signatures, we propose parallel coordinates attack visualization (PCAV). PCAV has two main modules, the analyzer and the visualizer as shown in Fig. 2. The analyzer receives the flow information from the sensor, and checks whether it contains a pattern matched with an attack signature. If a set of flows form a pattern matched with an attack signature, then it implies that the attack is currently ongoing. After that, the attack data is sent to the visualizer.

The visualizer displays flow data using parallel coordinates, where flow data can be obtained from the sensor or the analyzer. Flow data including both of legitimate and attack flows comes together from the sensor. But only attack data comes from the analyzer. We can store the attack data in a database for recording the attack and it can be useful for further investigation of the incident.

The sensor can be a host or a router which generates flow-level data from network traffic. A host can run a monitor program such as nProbe [7]. A router can be enabled to generate flow data such as NetFlow information in Cisco routers.

Rescaling properties of parallel coordinates can be used to magnify an attack graph. In absolute coordinates, the top and bottom values of each coordinate are fixed in constant. Thus, it has an advantage to estimate the region of the four parameters in an attack graph. However, the unfolded portion in a coordinate is too narrow to be recognizable

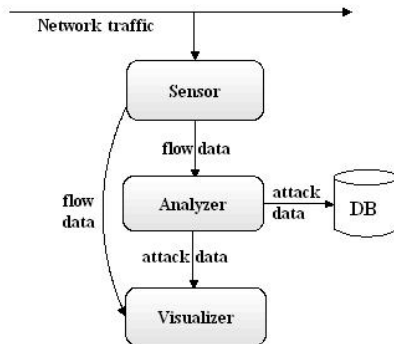


Fig. 2. System design of PCAV

by human, then, even if a detected attack was visualized, the attack graph may not show an obvious attack signature. Therefore, the visualizer in PCAV provides the rescaling operation by fitting the minimum and maximum value of each coordinates so that we can see an apparent attack signature in the attack graph.

### 3.3 Data Structure

In this subsection, we explain an attack detection algorithm which is running in the analyzer. The detection algorithm uses three hash tables for storing flows with respect to their source address, destination address and destination port, respectively. Then, the hash tables are used to determine which pattern in the attack signatures the flows have.

There are several data structures for storing flow information, which include linked lists, trees, and hash tables. A particular type of tree called MULTOPS [3] was also proposed for storing IP addresses efficiently. A comparison of hash tables with other structures such as linked lists, balanced binary trees, and MULTOPS trees, is shown in Table 2. From this comparison, we chose hash tables because they do not require huge memory space but provide fast lookup.

Fig. 3 shows the proposed attack detection algorithm. This algorithm consists of two parts. The first part is to generate a flow ID for an input flow using three hash tables, which is described at Step 1 ~ 12 in Fig. 3. The second part is to handle suspicious flows using the flow ID, which is described at Step 13 ~ 30 in Fig. 3. A flow ID has three tuples and a legitimate flow ID is either [0, 0, 0] or [1, 1, 1]. Otherwise, the flow is suspicious. If one flow comes from the sensor, the detection algorithm in the analyzer inserts the source address, destination address and destination port of the flow to each hash table. Then, the three hash tables are used to generate the three tuples of the flow ID. If an input value already exists in its hash table, then the tuple value becomes 1. Otherwise, becomes 0. For example, at time T1, if an input flow has a source IP address 1.2.3.4, destination address 5.6.7.8 and destination port 80, and at time T2, an input

**Table 2.** Comparisons of data structures

Algorithm	Pros	Cons	Complexity (lookup)
Linked List	Easy to implement Easy to use	High lookup complexity	$O(n^2)$
MULTOPS's tree data structure	4 lookup for searching (high speed)	1. Weak to Source spoofed DDoS attack (resource exhausted) 2. Memory usage (normal)	$O(1)$
Binary search tree (Balanced)	n level lookup memory usage	Lookup Complexity	$O(n \log n)$
Hash table	high speed	Computational overhead (function execution)	$O(1)$

```

Attack-Detect ( $F_n$ )  $F_n \leftarrow$  Flow data
1   $SA_n, DA_n, DP_n \leftarrow$  Source IP, Destination IP, Destination port of  $F_n$ 
2   $T_s, T_d, T_p \leftarrow$  Hash tables for  $SA_n, DA_n, DP_n$ 
3   $Attack\_ID \leftarrow$  0x0111 Initialize  $Attack\_ID$ 
4  IF hash_insert( $SA_n, T_s$ ) = TRUE  $\leftarrow SA_n$  is a new value of hash table  $T_s$ 
5       $Attack\_ID = Attack\_ID XOR$  0x0100
6  ENDIF
7  IF hash_insert( $DA_n, T_d$ ) = TRUE
8       $Attack\_ID = Attack\_ID XOR$  0x0010
9  ENDIF
10 IF hash_insert( $DP_n, T_p$ ) = TRUE
11      $Attack\_ID = Attack\_ID XOR$  0x0001
12 ENDIF
13 IF  $Attack\_ID =$  0x0011
14      $DDoS_{fx} \leftarrow F_n, DDoS_{fx} \leftarrow$  Temporary DDoS attack (fixed port) queue
15 ENDIF
16 IF  $Attack\_ID =$  0x0010
17      $DDoS_{vx} \leftarrow F_n, DDoS_{vx} \leftarrow$  Temporary DDoS attack (varied port) queue
18 ENDIF
19 IF  $Attack\_ID =$  0x0101
20     IF  $AvgLen_n > 48$   $AvgLen_n \leftarrow$  Average packet length of  $F_n$ 
21          $Worm \leftarrow F_n, Worm \leftarrow$  Temporary Worm queue
22     ELSE  $Hostscan \leftarrow F_n, Hostscan \leftarrow$  Temporary hostscan queue
23     ENDIF
24 ENDIF
25 IF  $Attack\_ID =$  0x0110
26      $Portscan \leftarrow F_n, Portscan \leftarrow$  Temporary portscan queue
27 ENDIF
28 IF  $Attack\_ID =$  0x0100
29      $Backscatter \leftarrow F_n, Backscatter \leftarrow$  Temporary backscatter queue
30 ENDIF
END of Attack-Detect

```

**Fig. 3.** Attack detection algorithm in the analyzer

flow has a source IP address 1.2.3.4, destination address 5.5.5.5 and destination port 21, then the second flow at T2 has a flow ID of [1, 0, 0]. If at time T3, an input flow has source IP address 3.4.5.6, destination address 5.6.7.8 and destination port 80, then its flow ID becomes [0, 1, 1].

There are 6 prepared queues for each detectable attack. Once the input flow is classified into a suspicious flow in the first phase, then the suspicious flow is inserted into an attack queue corresponding to the flow ID. For instance, if a flow ID is [0, 1, 1], then we insert it to the DDoS (port fixed) queue. If the size of an attack queue exceeds its threshold for a given period, then it is considered as the occurrence of the attack. Fig. 4 shows whole stages of the attack detection algorithm in the analyzer.

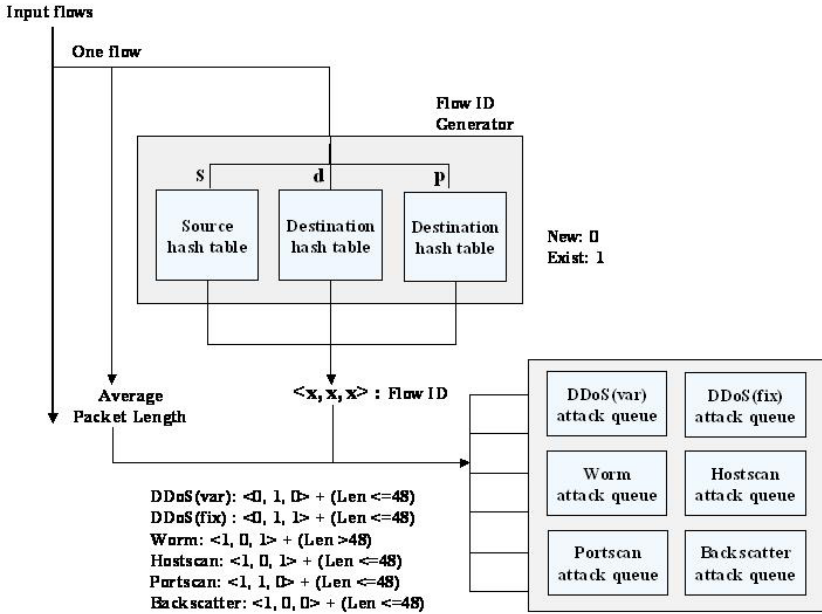


Fig. 4. Algorithm of PCAV

## 4 Evaluation

### 4.1 Attack Graphs

In order to evaluate the effectiveness of PCAV, we have implemented the PCAV system on Microsoft Windows. Attack situations are generated by replaying recorded real attack traffic. The attack traffic includes DoS attacks, SQL Slammer worms and network scanning traffic. DDoS attacks and SQL Slammer worms were captured during the incidents at one company network, and network scanning traffic was captured by using a public scanning tool. The PCAV system detects these attacks effectively and displays proper attack graphs as shown in Fig. 5.

All attack graphs are well-matched with the signature patterns shown in Table 1. The first graph is the DoS attack generated by a Blaster worm. Notice that the traffic is not a worm traffic but DoS traffic which generated by worm. The DoS attack uses a fixed destination port so the pattern represents a triangle with a connected line. The second graph is the attack graph by a Slammer worm, representing a noticeable pattern. The Slammer worm attempts to infect other machines chosen randomly so that the destination addresses should be in a random distribution. However, the pattern of the represented graph looks like a subnet scanning in the range of multicast IP addresses. This is due to a bug at the part of random number generation in the Slammer code, which generates only multicast address ranges in a certain condition. Even in this unusual situation, PCAV also detects the worm on the limited range of destination addresses.

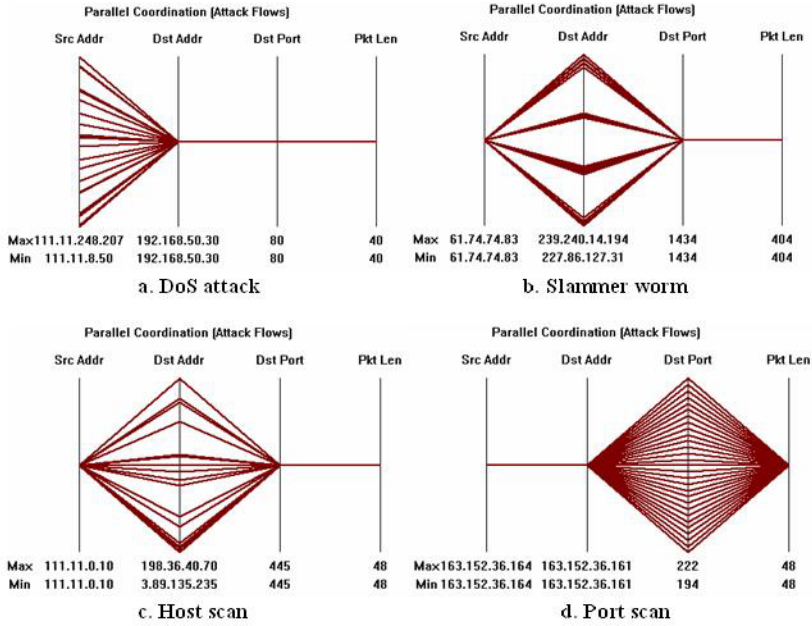


Fig. 5. Rescaled attack graphs

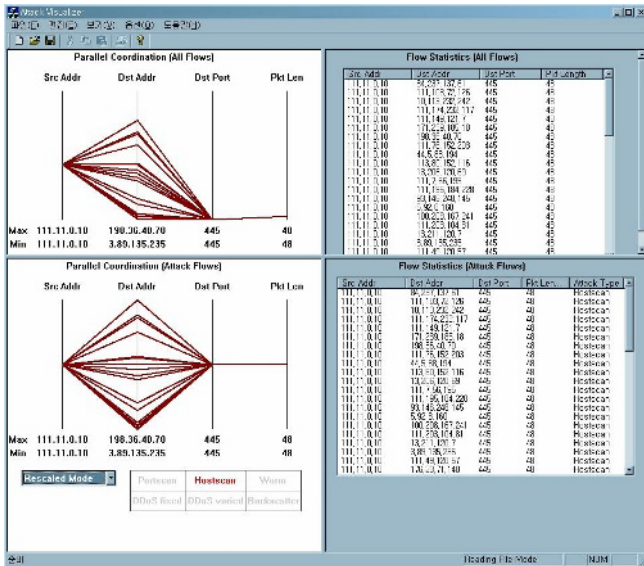


Fig. 6. Screenshot of PCAV 1.0

The third graph is generated by the PCAV system and it is the rescaled attack graph of the host scan in Fig. 1. The fourth graph is an attack graph of a port scan.

PCAV is implemented as a Windows application. PCAV can get flow data locally or import NetFlow data from remote routers. PCAV version 1.0 is shown in Fig. 6.

## 4.2 False Alarm Reduction

The rate of false alarm is an important metric used to measure the performance of an intrusion detection system. PCAV is sensitive to the threshold, which is closely related to the rate of false alarms. If a threshold is too high, then false positive will decrease but false negative will increase, and visa versa. Thus, we need to determine a proper threshold with the consideration of the size and the bandwidth of a monitoring network. Larger networks need to have the higher threshold.

We can further reduce false alarms by using additional parameters such as the cumulative OR of TCP flags. As well, more parameters can be added to parallel coordinates if they can enhance the correctness of attack detection.

## 5 Related Work

There are a number of popular monitoring tools such as FlowScan [8] and AutoFocus [9], used as traffic analyzers. Flowscan is a open source software that analyzes Net-Flow data and provides visualization graphs over five-minute intervals. AutoFocus automatically clusters traffic flows and infers patterns from the traffic.

In addition, there are several academic and commercial attempts to bridge information visualization to the field of intrusion detection. However, few of them provide real-time visualization and analyzing functionalities. One related work, regarded as previous work of this research, is 3-D visualization using a source address, destination address, destination port for detecting scanning and DDoS attacks [10]. But they cannot detect Internet worms properly and unable to distinguish legitimate traffic, such as P2P communications, from attacks.

Ourmon, which is developed at Penn State University, is an anomaly detection system using TCP flags and traffic volume as visualization sources. Ourmon presents simple bar graphs, however PCAV provides graphs which can be much more intuitively understood. Mazu Network's ProfilerTM [11], a commercial product, uses graphical profiling ability for network traffic in terms of IP address, protocols, ports, and flow volume.

Parallel coordinates are also used in other studies such as SHADOW [12], which was created at the Dahlgren Division of the Naval Surface Warfare Center. Packet headers meeting certain pre-defined boolean rules are dumped to a web-based file for examination by a human operator. SHADOW provides two visualization methods of colored histograms, parallel coordinates, and clustering methods are also used to solve various intrusion detection problems.

## 6 Conclusion

PCAV is a real-time visualization system for anomaly detection of Internet attacks. PCAV visualizes Internet attacks using four header fields (source IP address, destina-

tion IP address, destination port, packet length) from a flow and displays on parallel coordinates. Attack graphs generated from PCAV have specific patterns because the visual nature of the generated graphs is specific to each attack. PCAV enables the network administrator to rapidly detect and respond to malicious attacks. Even though an unknown attack may occur, specific characteristics are represented visually, and are detectable by PCAV. A plan to adopt pattern recognition methods in computer graphics areas, recognizing signatures generated by PCAV, is currently being designed. Also, visualization research regarding Spam mail distributions, P2P traffics, botnets and new types of DDoS and worm attack, is currently being undertaken.

## References

1. A. Inselberg: The plane with parallel coordinates. *The Visual Computer* 1(1985) 69–91
2. Information visualization resources, <http://www.infovis.org>
3. T. Gil, M. Poletto.: MULTOPS: a data-structure for bandwidth attack detection. *USENIX Security Symposium* (2001)
4. D. Keim.: Visual exploration of large databases. *Communications of the ACM* (2001) 38–44
5. Cisco NetFlow, <http://www.cisco.com/warp/public/732/Tech/netflow>
6. S. Axelsson.: Visualization for intrusion detection: Hooking the worm. *ESORICS* (2003)
7. nProbe, <http://www.ntop.org/nProbe.html>
8. D. Plonka.: Flowscan: A Network Traffic Flow Reporting and Visualization Tool. *USENIX LISA* (2000)
9. C.Estan, S.Savage and G.Varghese.: Automatically Inferring Patterns of Resource Consumption in Network Traffic. *ACM SIGCOMM* (2003)
10. H. Kim, I. Kang, and S. Bahk.: Real-time Visualization of Network Attacks on High-speed Link. *IEEE Network Magazine* (2004)
11. Mazu Network Profiler, <http://www.mazunetwork.com>
12. J. L. Solka, D. L. Marchette, and B. Wallet.: Statistical visualization methods for intrusion detection. *Computing Science and Statistics* (2000)

# Implementation of Packet Filter Configurations Anomaly Detection System with SIERRA

Yi Yin, R.S. Bhuvaneshwaran, Yoshiaki Katayama, and Naohisa Takahashi

Department of Computer Science and Engineering Graduate School of Engineering,  
Nagoya Institute of Technology,  
Gokiso, Showa-ku, Nagoya, 466-8555, Japan  
{yinyi, bhuvan, katayama, naohisa}@moss.elcom.nitech.ac.jp

**Abstract.** Packet filtering in a firewall is one of the useful tools for network security. Packet filtering examines network packet and decides whether to accept, or deny it and this decision is determined by a packet filtering configuration developed by the network administrator. An administrator may find hard to understand and maintain a configuration, and this burden will furthermore be increased to find anomalies between two configurations, especially when the size of filters in a configuration increased. This difficulty may leave the administrator with less confidence that the configurations are correctly and completely implemented. This paper presents a system with SIERRA (A systolic filter sieve array) which can detect the anomalies between two configurations. It provides three functions, side-effects analysis function, equality judgment function, and composition analysis function. Experimental results show that the proposed system is suitable for small network and configurations with large number of filters.

**Keywords:** Network security, Packet filtering, configuration, filter, anomaly detection.

## 1 Introduction

Network security has gained significant attention in recent years. Firewalls have become important integrated elements in our society. Packet filtering in firewall is used as a tool for improving network security and performance.

According to the configurations developed by the administrator, packet filtering is a decision of acceptance or denial of a packet. It is difficult for an administrator to understand and maintain configurations in different cases, such as to modify filters in configurations, or to replace one configuration with another configuration written in different language, or to find whether there exist redundant filters in hierarchical structure configurations.

The administrator always has to wonder if the configurations are really accomplishing what was intended, or if the configuration has some inadvertent hole in it that the administrator has somehow overlooked.

In this paper, we proposed an anomaly detection system to help administer to yield correct configurations with greater confidence.



We have used four primitive operations based on SIERRA (A Systolic Filter Sieve Array used for high-speed packet classification[1]) to implement the proposed system. In our paper, we use SIERRA as the data structure mainly to explain and compare two configurations[6]. The anomaly detection system has three functions as follows and we will discuss in section 4 in detail:

1. Side effects Analysis function.
2. Equality Judgment function.
3. Composition Analysis function.

The rest of the paper is organized as follows. In section 2, we introduce some background work of firewall and configuration. In section 3, we present about SIERRA, and explain the implementation steps of SIERRA. In section 4, we describe primitive operations and discuss three functions in detail. In Section 5, we first describe the experimental system, and findings. We discuss the related work in section 6. Finally, in section 7, we show our conclusions.

## 2 Background

A firewall is an intelligent device based on configurations between two or more networks for security purposes. A firewall configuration is a list of ordered filter-set that define the actions performed on network packets based on specific conditions. A filter is composed of some key fields (we called these key fields as predicates) and an action field.

The predicates of a filter represent the possible values of the corresponding fields in actual network packet that matches this filter. Each predicate could be a single value or range of values. The most commonly used predicates are: protocol type, source IP addresses, destination IP addresses, source port number, destination port number. The actions of a filter are either "accept" or "deny".

The packet is permitted or blocked by a specific filter if the packet header information matches all the predicates of this filter. Otherwise, the following filter is examined and the process is repeated until a matching filter is found, or else, the default filter action is performed. An example of a typical firewall configuration is shown in Fig.1.

	Protocol	SrcIP	DesIP	SrcPort	DesPort	Action
1:	top	*	123.4.5.6	>1023	23	Accept
2:	tcp	*	123.4.5.*	>1023	25	Accept
3:	top	129.6.48.254	123.4.5.9	>1023	119	Accept
4:	udp	*	123.4.**	>1023	123	Accept
5:	top	*	*	*	*	Deny

Fig. 1. Configuration example

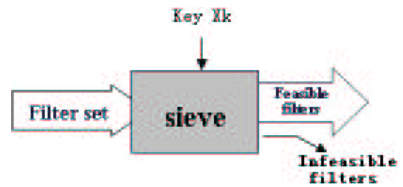


Fig. 2. Sieve function

### 3 SIERRA

#### 3.1 Problem of Point-Location

SIERRA (A Systolic filter Sieve Array) is used in a high-speed packet classifier. A packet classifier that deals with  $n$  key fields (header fields) is modeled as an  $n$ -dimensional point location problem in computational geometry. The  $n$ -dimensional point location problem is described as follows: given a point in  $n$ -dimensional space, and a set of  $m$   $n$ -dimensional objects, find the object that the point belongs to. In a packet classifier, a packet corresponds to the point and  $m$  filters correspond to the  $m$  objects.

#### 3.2 Sieve

Sieve is a function that can reduce an  $n$ -dimensional point-location problem to an  $n-1$  dimensional problem[1].

Given a filter-set  $F$ , sieve according to  $k$ -th ( $k=1,2,..,n$ ) key field value  $x_k$  of a packet to remove infeasible filters and return feasible filters. Infeasible filters are filters whose predicates corresponding to the key fields are false, and feasible filters are filters whose predicates corresponding to the key fields are true. Sieve is represented by  $sieve(F, k, x_k)$  and the sieve function is shown as in Fig.2.

#### 3.3 Structure of SIERRA

Fig.3 shows that if all the key fields are given as inputs in a pipelined sieve array, a filter-set in which predicates for all key fields are true is derived, that is, the feasible filter-set  $F'$  as the output of the final stage of the pipeline. This is represented as:

$$\begin{aligned}
 F' &= F_{n-1} \text{ provided that} \\
 F_0 &= sieve(F, 0, x_0), \\
 F_1 &= sieve(F_0, 1, x_1), \dots, \\
 F_{n-1} &= sieve(F_{n-2}, n-1, x_{n-1}),
 \end{aligned}$$

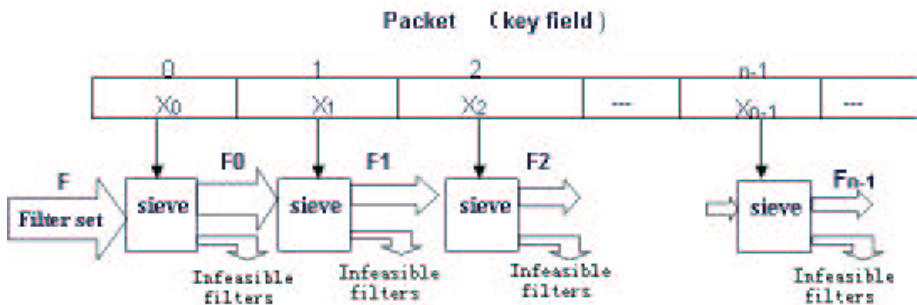


Fig. 3. A sieve array

### 3.4 Implementation of SIERRA

We use table (that we called sieve table) to implement SIERRA. We consider an abstract filter-set is shown below,  $F = \{f_1, f_2, \dots, f_m\}$

$$\begin{aligned}
 f_1 &: p_{1,1} \ p_{1,2} \ \cdots \ p_{1,n} \\
 f_2 &: p_{2,1} \ p_{2,2} \ \cdots \ p_{2,n} \\
 &\vdots \\
 f_f &: p_{f,1} \ p_{f,2} \ \cdots \ p_{f,n} \\
 &\vdots \\
 f_m &: p_{m,1} \ p_{m,2} \ \cdots \ p_{m,n}
 \end{aligned}$$

Here,  $f_1, f_2, \dots, f_m$  are filter identifiers, and  $p_{f,i}$  ( $f=1, \dots, m$  and  $i=1, \dots, n$ ) is the  $i$ -th predicate of filter. In order to make sieve table, we need several steps shown as follows:

**Step1: Represent the  $i$ -th predicate  $p_{f,i}$  ( $f=1, \dots, m$  and  $i=1, \dots, n$ ) of all the filters by 256-bit vectors whose elements are either T(true) or F(false).**

In our system, we define that the each predicate value, say  $x_i$  is 8-bit long ( $0 \leq x_i \leq 255$ ). Any point in this vector whose value changes from "T" to "F" or from "F" to "T" is called a boundary.

**Step2: Partition the domain at all boundaries and make domain descriptor at each sub-domain.**

The domain of the predicate value ( $0 \leq x_i \leq 255$ ) is partitioned into intervals at all boundaries of all predicates. An order set of values within each interval is called a sub-domain. The set of sub-domains has the following properties.

1. Disjoint: There are no pairs of sub-domains that have common elements.
2. Direct sum: The union of all sub-domains equal to the original domain.
3. Unique: When a sub-domain is determined, the filter-sets for which a "T" predicates is given are uniquely determined.

Each sub-domain is assigned an identifier number, which is called domain descriptor. Whenever there is a boundary, the domain descriptor is increased.

**Step3: Find the feasible filters at each sub-domain.**

**Step4: Check whether  $i$  is equal to  $n$ ; if  $i \neq n$ , according to feasible filters that get from Step3, execute Step1 through Step4 recursively.**

For example, we have a filter-set is shown as Fig.4. For simplicity, the filter-set have two filters and the predicate value of each filter is three bits long, that is the predicate value range is  $0 \leq x_i \leq 7$ , and can use a 8-bit vectors to present the predicate value. According to step1, we represent the first predicate of all filters into bit vectors. The first predicate  $p_{f,1}$  of all filters is shown in Fig.5. The bit vectors of the first predicate of all filters is shown as in Fig.6.

According to step2, we partition the domain of Fig.6 into 5 sub-domains shown in Fig.7 and each domain gives a domain descriptor.

f1: 1 ≤ SrcIP ≤ 5    1 ≤ DesIP ≤ 4  
 f2: 3 ≤ SrcIP ≤ 6    3 ≤ DesIP ≤ 6

Fig. 4. Example filter-set

f1: 1 ≤ SrcIP ≤ 5  
 f2: 3 ≤ SrcIP ≤ 6

Fig. 5. The first predicate of all filters in Fig.4

SrcIP value: 0 1 2 3 4 5 6 7  
 f1    F T T T T T F F  
 f2    F F F T T T T F

Fig. 6. Bit vectors of Fig.5

SrcIP value:	0	1	2	3	4	5	6	7
f1	F	T	T	T	T	T	F	F
f2	F	F	F	T	T	T	T	F
Domain-Descriptor	0	1	1	2	2	2	3	4

Fig. 7. Domain partition of Fig.6

SrcIP value:	0	1	2	3	4	5	6	7
f1	F	T	T	T	T	T	F	F
f2	F	F	F	T	T	T	T	F
Domain-Descriptor	0	1	1	2	2	2	3	4
Feasible								
Filter-sets	{ }	{f1}	{f1}	{f1,f2}	{f1,f2}	{f1,f2}	{f2}	{ }

Fig. 8. Make feasible filters of Fig.7

In step3, according to "T" or "F", we can find the feasible filters of all sub-domains. For example, when the SrcIP value is "3", at this column in Fig.7, the bit vector's element of filter "f1" and "f2" are all "T", so when SrcIP value is "3", the feasible filters are {f1,f2}. In this example, the feasible filter sets of all sub-domains are shown at the last line in Fig.8. According to step1 through step3, we can get sieve table of SrcIP in Table1. In step4, according to the feasible filters, we make the sieve table about next predicate to all the sub-domains. For example, in the third sub-domain (Domain descriptor=2), since the feasible filters are {f1,f2}, so we take the next predicate value of the feasible filters, and shown as follows.

$$f1: 1 \leq DesIP \leq 4$$

$$f2: 3 \leq DesIP \leq 6$$

Table 1. Sieve table of SrcIP

SrcIP value	Domain descriptor	Feasible Filters
0	0	{ }
1	1	{f1}
2	1	{f1}
3	2	{f1,f2}
4	2	{f1,f2}
5	2	{f1,f2}
6	3	{f2}
7	4	{ }

Table 2. Sieve table of DesIP when Domain descriptor=2 in Table1

DesIP value	Domain descriptor	Feasible Filters
0	0	{ }
1	1	{f1}
2	1	{f1}
3	2	{f1,f2}
4	2	{f1,f2}
5	3	{f2}
6	3	{f2}
7	4	{ }

According to step1 through step3 again, we can get the sieve table of DesIP in Table 2, like this, we make the next predicate's sieve table for all sub-domains, and for the space is limited, we don't list all the sieve tables.

## 4 Proposed System

The proposed system uses four primitive operations based on SIERRA. In this section, we first introduce the four primitive operations, and then explain three functions in detail.

### 4.1 Primitive Operations

Create, merge, mark and list-up are the four primitive operations which are discussed below.

**Create Operation.** Create operation can create sieve table for a given configuration according to the implementation steps of SIERRA discussed in section 3.4.

That is to say, we use sieve table to explain the meaning of configuration.

For example, let us consider two configurations, Configuration A and Configuration B, shown in Fig.9 and Fig.10. For simplicity, each configuration has "DesIP" and "Port" number. A part of sieve table of each configuration is shown in Fig.11 and Fig.12.

DesIP	Port	Action
A1: 10.0.0.1	S3	Accept
A2: 10.0.0.2	*	Deny
A3: 10.0.0.0	123	Accept
A4: *	*	Deny

DesIP	Port	Action
B1: 10.0.0.1	S3	Accept
B2: 10.0.0.2	177	Accept
B3: 10.0.0.0	123	Accept
B4: *	*	Deny

DesIP	Feasible filters	Port	Feasible filters
10.0.0.0	{A3A4}	0	{A2A4}
10.0.0.1	{A1A4}	...	255 {A2A4}
10.0.0.2	{A2A4}		
10.0.0.3-255	{A4}		

(a) sieve table of "DesIP"      (b) sieve table of "Port" when DesIP=10.0.0.2

Fig. 9. Configuration A      Fig. 10. Configuration B

Fig. 11. Sieve table of Fig.9

**Merge Operation.** If we have two sieve tables (sieve1 and sieve2), we merge two sieve tables into a new one. A new boundary is made in the merged sieve table where there is a boundary in sieve1 or sieve2. In the merged sieve table, the decision of feasible filters is according to the feasible filters in sieve1 and sieve2. For example, the merged sieve table of the first predicate "DesIP" is shown as in Fig.13 (a).

For example, when DesIP value is 10.0.0.0, the feasible filters in configuration A are {A3,A4}, while in configuration B are {B3,B4}, hence, when DesIP=10.0.0.0 in the merged sieve table, the feasible filters are{{A3,A4}{B3,B4}}.

DesIP	Feasible filters	Port	Feasible filters
10.0.0.0	{B3B4}	0	{B4}
10.0.0.1	{B1B4}	...	...
10.0.0.2	{B2B4}	176	{B4}
10.0.0.3-255	{B4}	177	{B2B4}
		178	{B4}
		...	...
		255	{B4}

(a) sieve table of "DesIP"

(b) sieve table of "Port" when DesIP=10.0.0.2

Fig. 12. Sieve table of Fig.10

DesIP	Feasible filters	Port	Feasible filters
10.0.0.0	{A3A4} {B3B4}	0	{A2A4} {B4}
10.0.0.1	{A1A4} {B1B4}	...	...
10.0.0.2	{A2A4} {B2B4}	176	{A2A4} {B4}
10.0.0.3-255	{A4} {B4}	177	{A2A4} {B2B4} *
		178	{A2A4} {B4}
		...	...
		255	{A2A4} {B4}

(a) Merged sieve table of Fig.12(a) and Fig.13(a)

(b) Merged sieve table of Fig.12(b) and Fig.13(b)

Fig. 13. Merged sieve table of Fig.11 and Fig.12

**Mark Operation.** In the merged sieve table of the last predicate, to each predicate value, we have feasible filters of two configurations, we take the highest priority filter of each configuration from feasible filters, and then compare the actions of the two filters. If the actions are the same, we do nothing, or else, the place with different actions between two configurations is made a mark.

For example, When the port number is 177, the feasible filters are {A2,A4} and {B2,B4}, the highest priority filter of configuration A is "A2", and the highest priority filter of configuration B is "B2", the action of A2 is "deny", while the action of B2 is "accept", hence, at this place, two configurations have different setting, and we make a mark "\*" at the place where the port number is 177 in Fig.13(b).

**List-up Operation.** The operation that list all marked places from the first predicate value to the last predicate value is called list-up operation.

For example, according to the marked place in Fig.13(b), we can list the predicate value range from the first predicate to the last predicate, it is shown as below.

DesIP	Port
10.0.0.2	177

## 4.2 Function of Proposed System

The proposed system has three functions which are discussed below:

**Side Effects Analysis.** When an administrator changed (add, delete, replace, or modify) filters in a configuration, changed filters may cause some potential problems, that is, a changed filter can cause side effects to other filters in configuration.

The side effects analysis function wants to find this kind of potential problems that caused by the changed filters. We present three pieces information as the output of this function.

1. The range of predicate value that influenced by the changed filter.
2. The action setting of each configuration to the range mentioned in 1.
3. Display the changed filters and the influenced filters in the configuration.

According to the result of this function, the administrator is able to master the influence caused by the changed filter, and correctly change the configurations according to the administrator’s intents.

For example, we have a configuration in Fig.14, and when the administrator wishes to add a piece of filter (shown as below), the original configuration as file1 and the changed configuration as file2, are considered as input of the proposed system, and the result is shown in Fig.15:

Type	SrcIP	DesIP	SrcPort	DesPort	Action
tcp	*	123.4.5.7	≥ 1023	25	Deny

Type	SrcIP	DesIP	SrcPort	DesPort	Action	
A1	tcp	*	123.4.5.6	>1023	23	Accept
A2	tcp	*	123.4.5.*	>1023	25	Accept
A3	tcp	129.6.48.254	123.4.5.9	>1023	119	Accept
A4	udp	*	123.4.**	>1023	123	Accept
A5	tcp	*	*	*	*	Deny

Fig. 14. Configuration

Anomaly :

Range of Influenced Predicate value :

Type	SrcIP	DesIP	SrcPort	DesPort
tcp	*	123.4.5.7	>1023	25

The action setting of influenced range in File1: Accept

The action setting of influenced range in File2: Deny

---

The influenced filters are:

tcp	*	123.4.5.*	>1023	25	Accept
-----	---	-----------	-------	----	--------

The changed filters are:

tcp	*	123.4.5.7	>1023	25	Deny
-----	---	-----------	-------	----	------

Fig. 15. Result of side effect analysis for the configuration shown in Fig.14

From the result, the administrator noticed that if he wants to add a filter, he should add this filter before A2 in the original configuration, or else, the added filter is no meaning, or the added filter can’t take its effect to packet.

We can implement this function using four primitive operations that introduced in section 4.1, and the implementation steps are shown as follows:

- Step1: Make sieve table of configuration. (Create operation)
- Step2: Make sieve table of the changed configuration. (Create operation)
- Step3: Merge two sieve tables get from step1 and step2 and seek feasible filters. (Merge operation)
- Step4: Compare two configurations and make a mark at the places where exist different setting. (Mark operation)
- Step5: Get the range of predicate value and the filters corresponding to the extracted place as the side effect analysis result. (List-up operation)

**Equality Judgment.** Configurations can be written in different description languages, for example, the iptables of Linux system, the access-list of cisco router, and etc. If the administrator wants to replace one configuration with

another configuration written in different language, the administrator always wonder whether the two configurations have the same meaning.

The equality judgment function can judge whether two configurations have the same meaning, even if two configurations are written in different description languages.

The equality judgment function can present three pieces of information as follows:

1. The range of predicate value that with different setting between two configurations.
2. The action setting in each configuration to the range mentioned in 1.
3. Display filters that caused the difference between two configurations.

Using the above information the administrator is able to detect the mistakes by the replaced device with different description languages.

For example, when the administrator wants to replace the iptables (Fig.16) by access-list of Cisco router (Fig.17), the proposed system can find whether there exist the different places between two configurations.

```
A1: iptables -A INPUT -p tcp -s 0.0.0.0 -d 0.0.0.0 -j ACCEPT
A2: iptables -A OUTPUT -i eth0 -p tcp --tcpflags ACK,ACK -j ACCEPT
A3: iptables -A OUTPUT -i eth0 -p tcp --syn -s 0.0.0.0 -d 0.0.0.0 -j ACCEPT
A4: iptables -A INPUT -j DROP
.....
```

Fig. 16. Configuration1(iptables)

```
B1: access-list 100 permit tcp any any range 137 140
B2: access-list 200 permit tcp any any established
B3: access-list 200 permit tcp any any syn
B4: access-list 200 deny ip any any
.....
```

Fig. 17. Configuration2(access-list)

The analysis result is shown in Fig.18:

```
Anomaly :
Range of Predicate value with Different setting :
Type  SrcIP  DesIP  SrcPort  DesPort
tcp   *      *      140      *
The action setting of this range in File1: Deny
The action setting of this range in File2: Accept
-----
The related filters in File1:
A2: iptables -A INPUT -j DROP
The related filters in File2:
B1: access-list 100 permit tcp any any range 137 140
B2: access-list 200 deny ip any any
```

Fig. 18. Result of equality judgment for configurations shown in Fig16 and Fig17



According to this result, if administrator wants to make configuration2's setting the same as to configuration1, a rule can be added (shown as below, for simplicity we only show the rule with simple style) before B1.

Type	SrcIP	DesIP	SrcPort	DesPort	Action
tcp	*	*	140	*	Deny

And if administrator wants to make configuration1's setting the same as to configuration2, a rule can be added before A2 (shown as below).

Type	SrcIP	DesIP	SrcPort	DesPort	Action
tcp	*	*	140	*	Accept

We can implement this function using four primitive operations, and the implementation steps are shown as follows:

- Step1: Make sieve table of one configuration. (Create operation)
- Step2: Make sieve table of another configuration. (Create operation)
- Step3: Perform merge operation like Step3 in function1. (Merge operation)
- Step4: Perform mark operation like Step4 in function1. (Mark operation)
- Step5: Get the range of predicate value and filters of extracted place with different settings as an analysis result of equivalence judgment. (List-up operation)

**Composition Analysis.** In hierarchical structure configurations, each level has a configuration with its own set of filters. Hence, there is a possibility of contradictory or redundant filters among lower and higher levels, which is highly difficult to verify by an administrator.

The composition analysis function is used to find whether contradiction or redundant filters are existing between two configurations that are successively used to filter a packet. This composition analysis function can present three pieces of information as follows:

1. Display the range of predicate value with contradiction setting.
2. Analyze and display the action of contradictive range in each configuration.
3. Display the filter pairs in each configuration that caused the contradiction.

Based on this information, the administrator can able to recognize the unforeseen problem such as the packet that wanted to pass through the lower-class firewall that is abandoned by the upper-class firewall.

For example: We have the upper-class configuration in Fig.19 and the lower-class configuration in Fig.20, the analysis output is shown as in Fig.21.

In this example, according to this result, the administrator can notice that at the range of

Type	SrcIP	DesIP	Port
udp	*	10.0.0.2	117

the action to all the packets in that range in upper-class configuration is "deny", while in lower-class configuration is "accept", and at the same time the administrator can find the filters in upper-class and lower-class configurations that caused the problem.

```
A1: permit udp any 10.0.0.1 eq 53
A2: deny udp any 10.0.0.2
A3: permit udp any 10.0.0.0 eq 123
A4: deny ip any any
```

Fig. 19. Upper-class Configuration

```
B1: permit udp any 10.0.0.1 eq 53
B2: permit udp any 10.0.0.0 eq 123
B3: permit udp any 10.0.0.2 eq 177
B4: deny ip any any
```

Fig. 20. Lower-class Configuration

```
Anomaly :
Predicate value Range with Contradict setting :
Type SrcIP DestIP Port
udp * 10.0.0.2 177
The action of the contradict range in File1: Deny
The action of the contradict range in File2: Accept
-----
The related filters in file1 are:
A2 deny udp any 10.0.0.2
The related filters in file2 are:
B3 permit udp any 10.0.0.2 eq 177
```

Fig. 21. Result of composition analysis for configuration shown in Fig.19 and Fig.20

We can implement this function using four primitive operations, and the implementation steps are shown as below:

Step1 ~ Step4: The same as Step1 ~ Step4 in function2.

Step5: From extract division chose the division that the upper-class configuration has "deny" setting while the lower-class configuration has "accept" setting.

Step6: Get the range of predicate value and filters corresponding to the place selected with Step5 as the analysis result of composition analysis. (List-up operation)

## 5 Experimental Evaluation

We implemented the proposed system with C program on a generic computer. In order to evaluate the feasibility and usability of our proposed system, we measure the synthesis time and memory usage of sieve table when we execute each function.

The synthesis time is the time of taken to create, merge, mark and list up operations. Memory usage of sieve table is the sieve table size that created by each function.

### 5.1 Experimental Items

We use two experimental items to measure the feasibility and the usability.

1. When the number of predicates increases, and the number of filters is fixed, measure the synthesis time and sieve table size of each function.
2. When the number of filters increases, and the number of predicates is fixed, measure the synthesis time and sieve table size of each function.

### 5.2 Test Data

We produce configurations with random filters in order to represent the realistic configurations. The filters number in a configuration are range from 5 to 30 for small network firewall, and the filters used in this experiment is that they be stateless.

Each filter can include at most 16 predicates (16-byte) shown as follow:

- Source and destination address(es): 8 byte
- Source and destination port number: 4 byte
- Protocol type: 1 byte
- Length: 2 byte
- TCP flags: 1 byte

In order to represent all kinds of configurations in practical firewall, in our experiment we select predicate number is equal to 6, 11, 16 to evaluate synthesis time and sieve table size of each function.

### 5.3 Experimental Results and Consideration

The experiment results are shown as in Table3 to Table 5.

From the result in Table3 to 5, we can see that even in the worst case (in Table4), the synthesis time and sieve table size are acceptable. In the worst case (in Table 4), the experimental system will take 0.97s ~ 4.51s and 1.45kb ~ 5.62kb memory size to detect anomalies between two configurations with filters from 5 ~ 30. And in the best case (in Table 5), we take 0.81s ~ 3.71s and 1.00kb ~ 3.18kb memory size to detect anomalies. Hence, our proposed system is feasible for small network firewall.

**Table 3.** The synthesis time and Sieve table size of Side effect

Width	6				11				16			
File1_num	5	12	20	30	5	12	20	30	5	12	20	30
File2_num	6	13	21	31	6	13	21	31	6	13	21	31
Time(sec)	0.82	1.39	2.04	2.70	1.54	2.40	2.91	3.27	2.17	3.09	3.42	3.98
Size(KB)	1.25	2.04	2.91	3.59	2.00	3.04	3.66	3.99	2.31	2.93	3.56	4.18

When the number of predicates is fixed, and the number of filters increases, the needed synthesis time and sieve table will increase, this kind of increase is called "increase A".

When the number of predicates increases, and the number filter is fixed, the needed synthesis time and sieve table will increase, this kind of increase is called "increase B".

Through Table3 to 5, we can find that "increase A" is smaller than "increase B". From this result we can confirm that only the predicates number is fixed, and the filter number increases, the synthesis time and sieve table size will not increase very quickly. Hence our proposed system can be usable for configurations with large number of filters.

**Table 4.** The synthesis time and Sieve table size of Equality Judgment

Width	6				11				16			
File1_num	5	10	20	30	5	10	20	30	5	12	20	30
File2_num	5	10	20	30	5	10	20	30	5	13	20	30
Time(sec)	0.97	1.10	0.99	1.65	1.49	2.35	2.60	3.49	2.07	2.70	4.06	4.51
Size(KB)	1.45	1.53	1.53	2.31	2.00	2.94	3.40	4.50	3.12	3.44	5.15	5.62

**Table 5.** The synthesis time and Sieve table size of Composition Analysis

Width	6				11				16			
File1_num	5	10	20	30	5	10	20	30	5	12	20	30
File2_num	5	10	20	30	5	10	20	30	5	13	20	30
Time(sec)	0.81	0.93	1.67	2.04	1.27	1.56	2.08	2.52	1.51	2.05	2.80	3.71
Size(KB)	1.00	1.31	1.93	2.56	1.62	1.93	2.56	3.18	1.62	1.93	2.56	3.18

## 6 Related Work

The similar work of ours has been done by Hazelhurst et al. [2]. The paper describes a method for transforming a firewall filter specified in a Cisco-like access list language into a BDD(Binary Decision Diagrams), including the handling of issues with overlapping rules. Although this work is capable of answering questions on the types of packets allowed or excluded by a set of firewall rules and even able to display the redundant rules between two configurations, it has many limitations as follows:

1. This tool is insufficient to represent the anomaly detection to various situations of modern firewalls, while in our proposed system, through three functions can provide anomaly detection in detail.
2. This tool has not efficiently implemented in software, while through the experiment result we had evaluate the feasibility and usability of our proposed system.

Another similar work has been done by Pasi Eronen and Jukka Zitting in an expert system for analyzing firewall rules [5]. The expert system is used for verifying the functionality of filtering rules by performing queries.

This work is based on the use of the principle of expert system where firewall access-lists are converted directly into an expert system knowledge base. The resulting knowledge base is then manipulated using the underlying inference engine of readily available Logic Programming language interpreters.

Although this research can address four main areas of problems that is network properties, configuration properties, access-list properties and defining new predicates for higher level queries, this research also has limitations as follows:

1. This tool does not provide comparison function between two configurations, while our proposed system can detect anomaly between two configurations in detail.

2. This tool requires high user expertise to write the proper queries to identify different configuration problems, while our proposed is simple to user to find anomaly easily.

## 7 Conclusion

This paper presents a system with SIERRA, used to detect anomalies among packet filter configurations.

This proposed system provides three functions, side-effects analysis function when modify a configuration, equality judgment function between two configurations written in different description languages, and the composition analysis function configurations in a hierarchical structure.

Experimental results show that although the synthesis time and sieve table size will increase when the filter number and predicate number increase, the proposed system is suitable for small network and useable for configurations with large number of filters.

**Acknowledgment.** This research was partially supported by the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for JSPS Fellows 1604285 and Scientific Research (C) 16500028.

## References

1. N.Takahashi, "A Systolic Sieve Array for Real-time Packet Classification," IPSJ Journal, Vol.42, No.2, pp.146-166(2001)
2. Soctt Hazelhurst, Anton Fatti,and Andrew Henwood. Binary decision diagram representations of firewall and router access lists. Technical Report TR-Wits-CS-1998-3, Department of Computer Science, University of the Witwatersrad, South Africa October 1998
3. Randal E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys, 24(3):293-318, 1992.
4. Lakshman,T.V.and Stiliads,D.:High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching, Proc.SIGCOMM 98,pp203-214 (1998).
5. P.Eronen and .J.Zitting. An expert system for analyzing firewall rules. In Proc.6th Nordic Workshop on Secure IT Systems(NordSec 2001), pages 100-107, Nov.2001.
6. Yin Yi, Yosshiaki katayama, Naohisa Takahashi. "A system for Comparing Packet Filter Configuration Files with SIERRA". 2004. Tokai branch rengo conference. (In Japanese)
7. Gupta, P. and Mckeown, N.:Packet classification on multiple fields, Proc. SIGCOMM 99, pp.147-160(1999).
8. Takahashi, N.: Real-time packet classification based on the partial evaluation of filter-sieve functions (in Japanese), Proc. Workshop on Internet Technologies 99, pp.190-197, JSSST (1999).

# D\_DIPS: An Intrusion Prevention System for Database Security

Jiazhu Dai and Huaikou Miao

College of Computer Science and Engineering,  
Shanghai University,  
Shanghai, P.R. China, 200072  
{daijz, hkmao}@staff.shu.edu.cn

**Abstract.** There is a growing security concern on the increasing number of databases that are accessible through the Internet because a variety of attacks do succeed to fool the existed database protection mechanisms in many applications. Defense-in-depth strategies like intrusion prevention is urgently needed for database security. Most of research on intrusion prevention focuses on preventing attacks on operating systems and computer networks. Few efforts have been put on database intrusion prevention. Design and implementation of a database intrusion prevention system D\_DIPS is presented. The goal of D\_DIPS is to detect attacks caused by malicious transactions and cancel them timely before they succeed. The D\_DIPS prototype shows D\_DIPS can detect and stop attacks of malicious transaction in real time with low false alarm rate.

## 1 Introduction

Database systems form the core of the information systems infrastructure in large organizations. These databases support a large variety of applications such as electronic commerce, management information systems, hospital information systems. Data stored in these databases ranges from personal information and commercial secrets to banking transactions and medical records. Any breach of security to these databases may cause losses for customers and organizations. Therefore, It is very important that the data stored in the database systems must be protected from unauthorized access and modification.

Although security mechanisms such as authentication, access control, inference control, encryption, multilevel secure databases have been deployed to protect database systems, a variety of attacks have fooled these security mechanisms and caused unauthorized access and modification of data stored in databases. One solution in face of these attacks is intrusion prevention techniques. Intrusion prevention is a proactive defense technique which is extension of intrusion detection. Intrusion prevention systems detect ongoing attacks in real time and stop the attacks before they succeed, thus, avoid damage caused by the attacks.

Most of research on intrusion prevention focuses on preventing attacks on operating systems and computer networks [1],[2],[3],[4],[5]. Although auditing tools are

provided by most of popular DBMS and there are some works on database intrusion detection[6],[7],[8],[9],[10],[11],[12],[13], these methods focus on detecting attacks after the event, so they can not realize intrusion prevention for database. To our best of knowledge, few efforts have been put on database intrusion prevention. Ulf T. Mattsson presented an intrusion prevention system for database in [14], [15],[16]. His researches focus on monitoring database objects (such as tables, attributes, etc) access rates associating each user, if the access rates exceed the threshold, notifying the access control system to make the user's request an unauthorized request before the result is transmitted to the user. Our work in this paper is different from that of Ulf T. Mattsson in that we focus on monitoring transactions rather than access rates, and proactive protection is based on atomicity of transactions rather than modification of users' authorization.

Design and implementation of database intrusion prevention system D\_DIPS is presented in this paper. The D\_DIPS focuses on detection and prevention of malicious transactions. Malicious transactions are transactions that access database without authorization, or transactions that are issued by users who are authorized but abuse their privileges. The D\_DIPS monitors transactions issued by users and malicious transactions are viewed as intrusion behaviors. If a malicious transaction is identified, the D\_DIPS cancel the transaction before it succeeds, thus minimize damage caused by malicious transactions. The paper is organized as follows: a brief review of intrusion detection and intrusion prevention is described in section 2. The next section presents intrusion prevention model of D\_DIPS. The architecture of D\_DIPS is presented in section 4. The intrusion detection in D\_DIPS is described in section 5. The implementation of D\_DIPS prototype and its performance is presented in section 6. We conclude our paper in section 7.

## 2 Intrusion Detection and Intrusion Prevention

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. Intrusions are caused by attackers who exploit vulnerabilities in hardware or software to gain unauthorized access or to exceed their privileges in the computer system.

Intrusion detection approaches can be divided into misuse detection, anomaly detection and specification-based detection. Misuse detection techniques detect attacks as instances of attack signatures. This approach can detect known attacks accurately, but is ineffective against previously unseen attacks, as no signatures are available for such attacks. Anomaly detection overcomes the limitation of misuse detection by focusing on normal system behaviors, rather than attack behaviors. The profiles of normal system behaviors are usually created by machine learning techniques. Any deviation from normal system behaviors is treated as potential attacks. Specification-based techniques detect attacks as deviations from a norm, which is similar to anomaly detection. However, instead of relying on machine learning techniques, specification-based approaches are based on manually developed specifications that capture legitimate (rather than previously seen) system behaviors. They avoid the high rate of

false alarms caused by legitimate-but- unseen-behavior in the anomaly detection approach. Their shortcoming is their time-consuming development of detailed specification.

Intrusion detection techniques make computer systems attack-aware but not attack-resistant, that is , intrusion detection itself can not maintain confidentiality and integrity of the computer system in face of attacks. Intrusion prevention techniques are extension of intrusion detection, they not only monitor events in computer systems to detect potential attacks, but timely stop the attacks when they are detected to minimize losses caused by the attacks.

In this paper, database intrusion prevention focuses on detecting and preventing application layer attacks which exist in the form of malicious transactions.

### 3 The Database Intrusion Prevention Model

The database intrusion prevention model of D\_DIPS is depicted in figure 1. The model is integration of intrusion detection and access control. The main idea of the model is that intrusion detection is passive defense-in-depth solution which is able to monitor users’ activities but unable to prevent unauthorized activities when they are detected., while access control is able to prevent users’ unauthorized access to database but unable to monitor users’ activities when access permits are granted. If we integrate these two technologies, we can overcome disadvantages of both technologies and provide more secure defense-in-depth protection for database security.

In order to realize application layer intrusion prevention ,the database intrusion prevention model should provide following functions:

- **Interception:** The database intrusion prevention system should intercept any transactions issued by client applications to access the database.
- **Analysis and decision:** The intercepted transactions’ information should be analyzed to determine whether they are malicious transactions.
- **Response:** If the analyzed transactions are identified as malicious transactions, the database intrusion prevention system should cancel them before they succeed, thus intrusion prevention is achieved.

The database intrusion prevention model consists of mediator and intrusion detector. The mediator acts as proxy between the clients and the database. It captures trans-

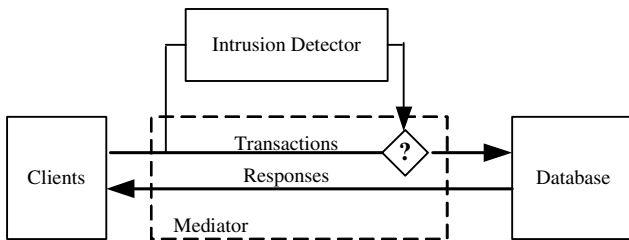


Fig. 1. The D\_DIPS Mode



actions' information for intrusion detector while proxying each users' transactions from the clients to the database. When intrusions are detected , the mediator responds to the intrusion alarms by canceling the malicious transactions before they are finished. The intrusion detector realizes analyzing function in the model. It analyzes transactions submitted from clients to database, if a malicious transaction is detected, the intrusion detector informs the mediator immediately to cancel the transaction before it succeeds. The diamond with question mark represents mediator passes or cancels a transaction according to intrusion detector's detecting results.

### 4 The Architecture of D\_DIPS

The deployment of D\_DIPS in database application is depicted in figure 2. There are following assumption about D\_DIPS:

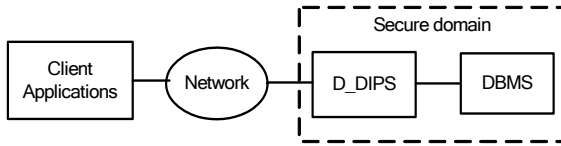


Fig. 2. The Deployment of D\_DIPS

- All database users must interact through D\_DIPS with database. There are no direct interactive ways for database users to bypass the D\_DIPS. That is , all users' accesses to database are monitored by D\_DIPS.
- All client applications access database through transactions generated by client applications and manipulating logic of these transactions is secret from attackers. This assumption satisfies typical database application cases as most database applications do not allow users issue their SQL queries/statements. Users typically specify their requirements through a client interface and SQL statements are generated by client applications.

The architecture of D\_DIPS is depicted in figure 3. The intrusion detector detects malicious transactions based on transaction trails at three levels: sessional level, schematic level and semantic level. Transactions are checked at the divided levels by D\_DIPS according to the preset detecting granularities. If a malicious transaction is detected at one of these three levels, the intrusion detector informs alarm module to send a alarm to security administrator and informs mediator to cancel the malicious transaction immediately before it succeeds. The transactions' trails and any information of malicious transactions is recorded by audit module for future analysis. The configuration module sets security rules, normal behaviors' profiles and detecting granularities for intrusion detector according to security policies.

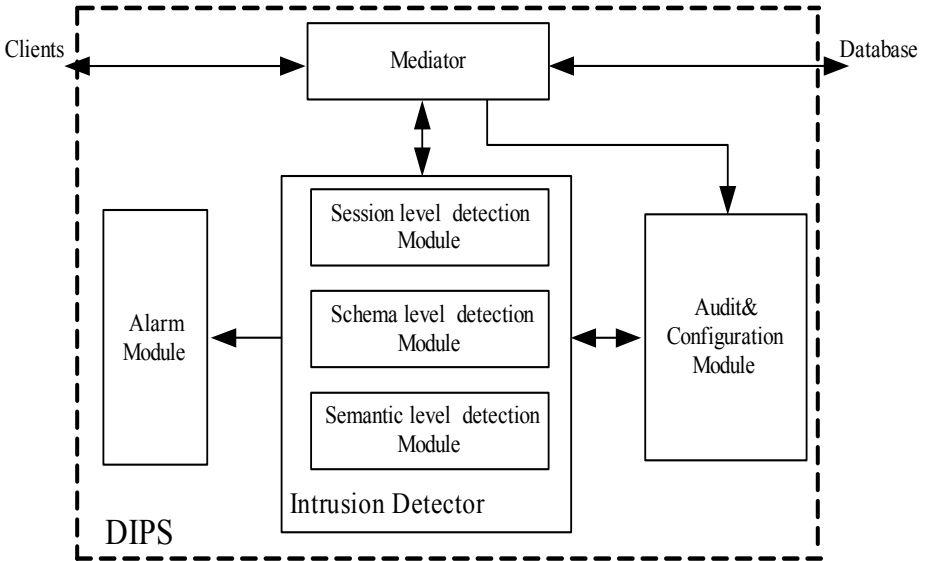


Fig. 3. Architecture of D\_DIPS

### 5 Intrusion Detection in D\_DIPS

The intrusion detector detects malicious transactions based on transaction trails at three levels: sessional level, schematic level and semantic level. Sessional level detection checks whether a transaction is authorized or not based on the validity of sessional level trails such as session time of the transaction , the database username or computer name issuing the transaction. Specification-based intrusion detection approach is used in sessional level detection. A lookup table containing legitimate behaviors' parameters(authorized user name, authorized computer name, valid accessing time period)like table 1 is maintained for sessional level detection. There are three columns in it. When a transaction is issued, the database username, computer name issuing the transaction and the session time of the transaction are compared with the corresponding columns in each records in table 1. Any deviations from these parameters or non-existence are treated as intrusion.

Table 1. The lookup table used for sessional level detection

Username	Computer Name	Valid time period
$U_1$	$C_1$	$T_{1s} - T_{2e}$
.	.	.
.	.	.
.	.	.
$U_n$	$C_n$	$T_{ns} - T_{ne}$

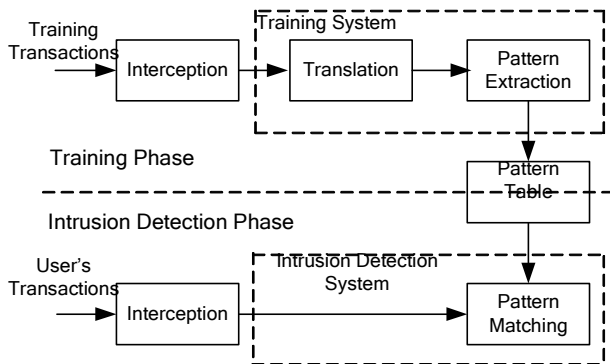
The schematic level detection is based on schematic information of SQL statements contained in transactions. The schematic level detection consists of two parts:

- Detecting whether SQL statements in transactions violate predefined access control policies. Specification-based intrusion detection approach is used which define authorized accessing privileges for every database user as access control matrix as table 2 shows. Users are denoted by  $U$ . Database objects such tables and attributes are denoted by  $O$ ,  $A(U_i, O_j)$  means authorized accessing privileges for user  $U_i$  to access database object  $O_j$ . Any violation of these privileges means attacks.

**Table 2.** Access Control Matrix

$U \backslash O$	$O_0$	$O_1$	$O_2$	$\dots$	$O_n$
$U_1$		$A(U_1, O_1)$	$A(U_1, O_2)$	$\dots$	$A(U_1, O_n)$
$\cdot$		$\cdot$	$\cdot$	$\cdot$	$\cdot$
$\cdot$		$\cdot$	$\cdot$	$\cdot$	$\cdot$
$\cdot$		$\cdot$	$\cdot$	$\cdot$	$\cdot$
$U_m$		$A(U_m, O_1)$	$A(U_m, O_2)$	$\dots$	$A(U_m, O_n)$

- Detecting whether schematic structure of every SQL statement in a transaction matches with corresponding part in profiles. Anomaly detection approach is used which consists of training phase and detecting phase as shown in figure 4.



**Fig. 4.** Intrusion detection based on schematic patterns of SQL statements

The intrusion detection based on schematic structures of SQL statements must be trained first in the training phase without existence of any intrusion behaviors. In the training phase, the users issue transactions accessing database through client application. The SQL statements in transactions are captured and transformed into regular expressions stored in pattern table. For example, the following SQL statement:

```
Select product_no,order_date
from Order
Where order_no="A123456" AND amount>10000
```

is transformed into following regular expression:

```
^Select product_no,order_date
from Order
Where order_no="[:alnum:]]+"AND amount>[[:digit:]]+$
```

In intrusion detection phase, SQL statements in users' transactions are intercepted and compared with corresponding regular expressions in the pattern table. Any mismatch indicates intrusion.

In current research, we assume schematic structures of every SQL statement in one transaction is invariable every time when the same transaction is issued by client application. For example, let following SQL statement be one in a transaction:

```
Select Username, Address, Balance
from Credit-card
where Accountnumber= "123456" AND Company= "ABC Company" AND
Address="Maple Street"
```

Every time when the same transaction is issued by client application, the structure of the SQL statement is fixed, only parameters in the condition part such as "123456", "ABC Company", "Maple Street" are variable, so following variable schematic structures of this example are not considered in current research:

```
Select Username, Balance
from Credit-card
where Accountnumber= "123456"
```

or

```
Select Address, Balance
from Address= "ABC Company"
```

As assumed previously, client users access database through transactions generated by client applications. So the schematic structures of SQL statements in transaction have following features under the above assumption:

- **Regularity:** Users can not create SQL statements by themselves. What they have to do is to specify their requirements through a client interface. SQL statements are generated by client applications according to the users' requirements. So such SQL statements are regular.
- **Stability:** The schematic structures of SQL statements in transactions generated by client application are stable unless client application is changed. This feature and the first feature help us reduce false positive alarm rate.
- **Limited:** Different transactions generated by client application are limited. This feature make it feasible for us to collect schematic structures' information of all SQL statements contained in different transactions in training phase to achieve low false negative alarm rate.

Semantic level detection is based on such trails as attributes values or statistical results of attributes in SQL statements. These trails indicate the operation semantics of SQL statements in transactions. Misuse detection is used in semantic level intrusion detection. Every SQL statement in a transaction is assigned a value called anomaly

measure value(AMV) during semantic level checking. If attributes values or statistical results of attributes in one SQL statement contained in a transaction match predefined detecting rules, that indicates a possible attack may be exist, a non-zero AMV is set for that SQL statement. Otherwise zero is set. If the sum of AMV of all SQL statements in a transaction is larger than preset threshold, that is, if the following equation holds:(let  $m$  be the number of SQL statements contained in the transaction and  $A_i$  be the AMV of  $i$ th SQL statement )

$$\sum_{1 \leq i \leq m} A_i \geq Threshold .$$

then the whole transaction is anomaly and actions should be taken to cancel it.

There are two reasons why transaction trails are divided into sessional level, schematic level and semantic level. The first reason is that we can make good use of transaction information at different levels to detect malicious transactions accurately. The second reason is that we can set different detecting granularity for different database objects to achieve trade-off between impact of D\_DIPS on database performance and database security. For example,we can set D\_DIPS to monitor transactions manipulating some trivial database tables only at sessional level for high performance and monitor transactions manipulating some important database tables at sessional level, schematic level and semantic level to achieve high security.

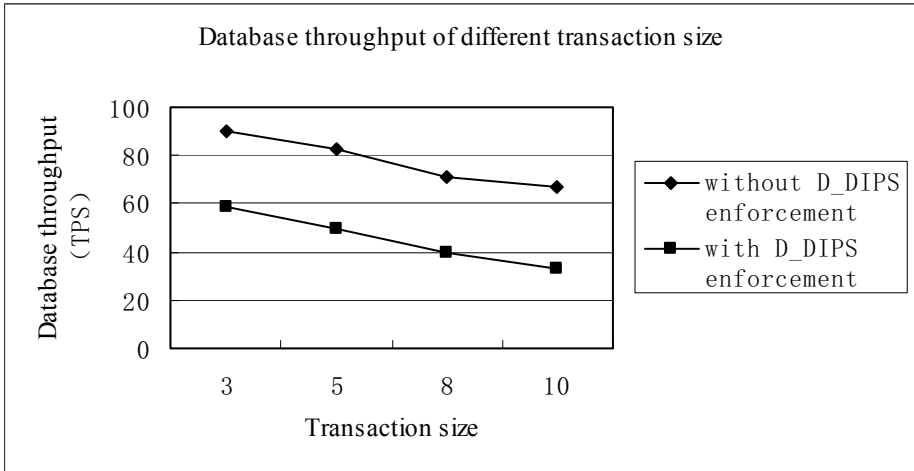
When a intrusion is detected. D\_DIPS cancel the transaction based on atomicity of transactions, that is, the attack of the intrusion can be stopped as follows: The SQL statements in a transaction except the last one indicating commit or rollback (COMMIT WORK, ROLLBACK WORK) of this transaction are proxied to DBMS by the mediator. Whether a transaction is committed or cancelled is determined by detecting results of intrusion detector. If a malicious transaction is identified, the mediator rollbacks the malicious transaction to cancel its operation, otherwise, the mediator commits the transaction.

## 6 Implementation and Performance

We have implemented a D\_DIPS prototype based on open source database PostgreSQL. The mediator proxies transactions from clients application to PostgreSQL through PostgreSQL client interface libpq.

One concern of deploying D\_DIPS is its impact on database performance. We test the D\_DIPS prototype in following testbed: the client machine is Intel PentiumIV 500Hz with 128MB of RAM, the machine running D\_DIPS is Intel PentiumIV 1GHz with 128MB of RAM ,the machine hosting database server is Intel PentiumIV 1GHz with 128MB of RAM. All the machines run redhat linux 7.0 .The database system is PostgreSQL 7.3.3.

We define transaction size is the number of SQL statements contained in a transaction. The performance of PostgreSQL is measured by throughput which is defined by the ratio of the number of transactions to the time to finish all the transactions. In testing environment with and without D\_DIPS enforcement ,1000 transactions with different size (size 3,5,8,10) are issued by client application to PostgreSQL respectively. The transactions are checked by D\_DIPS at sessional level, schematic level



**Fig. 5.** Database throughput with and without D\_DIPS enforcement

and semantic level. The database throughput with and without D\_DIPS enforcement is shown in figure 5.

The performance penalty caused by D\_DIPS is about 42% in average. But the result is measured under environment of checking every SQL statement in transactions at sessional level, schematic level and semantic level. We can achieve higher performance by setting different detecting granularities for transactions accessing different database tables.

Intrusion prevention effect of D\_DIPS is another concern when deploying D\_DIPS. As described previously, the characteristics we use to detect malicious transactions at sessional level, schematic level and semantic level is stable and regular. Therefore, in our experimental environment, the intrusion detector raises very few false alarms and malicious transactions are stopped by D\_DIPS before they succeed.

## 7 Conclusion

Defense-in-depth solutions are urgently needed in protecting confidentiality and integrity of data stored in database due to frequent reports on breach of database security. Database intrusion prevention is extension of database intrusion detection, which makes database attack-aware but not attack-resistant. Database intrusion prevention turns passive defense of intrusion detection into proactive defense through detecting and responding ongoing attacks in real time. Design and implementation of a database intrusion prevention system D\_DIPS is presented in this paper. The D\_DIPS prototype shows our D\_DIPS can detect and stop database application layer attacks caused by malicious transactions in real time with low false alarm, thus minimize loss caused by malicious transactions. The impact of D\_DIPS on database system can be minimized by suitable detecting granularity configuration. Our future research directions include (a) automatic summarizing schematic regular expression from variable sche-

matic structures of same SQL statement; (b) extending the D\_DIPS prototype from client/server architecture to three-tier architecture (c) Integrating database intrusion prevention techniques with COTS database products to improve their security.

## References

1. Cholter L W, Narasimhan P, Sterne D, Balupari R, Djahandari K, Mani A, Murphy S. IBAN: intrusion blocker based on active networks. Proceedings of DARPA Active Networks Conference and Exposition(2002) 182–192
2. Janakiraman R, Waldvogel M, Zhang Q. Indra: a peer-to-peer approach to network intrusion detection and prevention. Proceedings of Enabling Technologies: Infrastructure for Collaborative Enterprises(2003) 226–231
3. Ryutov T, Neuman C, Kim D, Li Z. Integrated access control and intrusion detection for web servers. IEEE transactions on parallel and distributed systems, 14(9), (2003) 841–850
4. Sekar R, Uppuluri P. Synthesizing fast intrusion prevention/detection system from high-level specifications. Proceedings of the 8<sup>th</sup> USENIX security symposium, Washington, D.C. (1999)
5. Stevens J, Saniepour S. SecureDirect: proactive security through content based traffic control. Proceedings of 17th International Conference on Advanced Information Networking and Applications(2003) 704–709
6. Ammann P, Jajodia S, McCollum C.D, et al. Surviving information warfare attacks on databases. Proceedings of the IEEE Symposium on Security and Privacy(1997) 164–174
7. Chung C Y, Gertz M, Levitt K. DEMIDS: A misuse detection system for database systems. Proceedings of the 3<sup>rd</sup> International IFIP TC-11 WG11.5 Working Conference on Integrity and Internal Control in Information Systems(1999) 159–178
8. Ingsriswang S, Liu P. AAID: An application aware transaction-level database intrusion detection system. Technical Report, Dept. of Information Systems, UMBC(2001)
9. Lee S Y, Low W L, Wong P Y. Learning fingerprints for a database intrusion detection system. ESORICS 2002, LNCS 2502(2002) 264–279.
10. Lee V, Stankovic J, Son S. Intrusion detection in real-time database systems via time signatures. Proceedings of the 6<sup>th</sup> IEEE Symposium on Real Time Technology and Applications(2000) 124–133
11. Low W L, Lee S Y, Teoh P. DIFAFIT: Detecting intrusion in databases through fingerprinting transactions. Proceedings of the 4<sup>th</sup> international conference on enterprise information system(ICEIS)(2002)
12. Shun W H, Daniel, T.T.H. A novel intrusion detection system model for securing web-based database systems. Proceedings - IEEE Computer Society's International Computer Software and Applications Conference(2001) 249–254
13. Stolfo S, Fan D, Lee W, et al. Credit card fraud detection using meta-learning: issues and initial results. Proceedings of AAAI Workshop: AI approach to fraud detection and risk management(1997)
14. Ulf T. Mattsson. A practical implementation of a real-time intrusion prevention system for commercial enterprise databases. Management Information Systems, v 10, Data Mining V: Data Mining, Text Mining and their Business Applications(2004) 263–272
15. Ulf T. Mattsson. A real-time intrusion prevention system for commercial enterprise databases and file systems. Proceedings of the Third IASTED International Conference on Communications, Internet, and Information Technology (2004) 189–194
16. Ulf T. Mattsson. A real-time intrusion prevention system for enterprise databases. [http://www.quest-pipelines.com/newsletter-v5/1104\\_B.htm](http://www.quest-pipelines.com/newsletter-v5/1104_B.htm)

# Author Index

- Ballard, Lucas 414  
Bao, Feng 40, 53, 207, 402  
Baratto, Ricardo 363  
Bhuvaneshwaran, R.S. 467  
Biryukov, Alex 147  
Boyd, Colin 84
- Cathalo, Julien 291  
Cederquist, J. 27  
Chen, Kefei 269  
Chiu, Yun-Peng 280  
Choi, Hyunsang 454  
Chow, Sherman S.M. 194  
Cook, Debra L. 363  
Corin, R. 27
- Dai, Jiazhu 481  
Dawson, Ed 84  
Deng, Robert H. 53, 207, 376  
Ding, Jintai 159  
Ding, Xuhua 269  
Dong, Shou-Ling 231  
Duan, Haixin 243
- Fellah, Alaaeddine 123  
Feng, Dengguo 14
- Goi, Bok-Min 136, 159
- Herrera-Joancomartí, Jordi 427  
Hu, Mingzeng 220  
Huang, Chun-Ying 280  
Huang, Song 231
- Imamoto, Kenji 1, 40  
Izu, Tetsuya 72
- Jalili, Rasool 256  
Ji, Zhenzhou 220  
Jia, Weijia 402  
Jiang, Xinghao 336
- Kamara, Seny 414  
Kanaya, Nobuyuki 72  
Katayama, Yoshiaki 467
- Keromytis, Angelos D. 363  
Kim, Jongsung 147  
Kim, Sangjin 323  
Kuribayashi, Minoru 441  
Kuwakado, Hidenori 112
- Laur, Sven 97  
Lee, Byoungcheon 84  
Lee, Heejo 454  
Lee, Hoonjung 323  
Lee, Sangjin 147  
Lei, Chin-Laung 280  
Li, Jianhua 336  
Li, Lan 336  
Li, Tieyan 389  
Li, Xin 220  
Li, Xing 243  
Li, Yong 61  
Libert, Benoît 291  
Lipmaa, Helger 61, 97
- Ma, Di 376  
Megías, David 427  
Miao, Huaikou 481  
Mielikäinen, Taneli 97  
Minguillón, Julià 427  
Mitchell, Chris J. 304  
Monrose, Fabian 414  
Morii, Masakatu 112, 441  
Mullins, John 123
- Nguyen, Khanh 181  
Ning, Peng 350
- Oh, Heekuck 323  
Omidian, Ali Reza 256
- Pang, Hweehwa 376  
Pei, Dingyi 61  
Peng, Kun 84  
Phan, Raphael C.-W. 136  
Preneel, Bart 147
- Qi, Fang 402  
Quisquater, Jean-Jacques 291
- Reeves, Douglas S. 350



- Sadoddin, Reza 256  
Sakai, Yasuyuki 169  
Sakurai, Kouichi 1, 40, 169  
Shahriari, Hamid Reza 256  
Siddiqi, M.U. 159  
Susilo, Willy 194
- Takahashi, Naohisa 467  
Takenaka, Masahiko 72  
Tanaka, Hatsukazu 112, 441  
Tang, Qiang 304  
Torabi Dashti, M. 27
- Wang, Guilin 40  
Wang, Lanjia 243  
Wang, Pan 350  
Wang, Shuhong 53
- Wei, Wei 269  
Wu, Yongdong 389, 402
- Xu, Jing 14
- Yin, Yi 467  
Yoo, Kee-Young 315  
Yoon, Eun-Jun 315  
Yoshioka, Takashi 72
- Zakeri, Reza 256  
Zhang, Ling 231  
Zhang, Zhenfeng 14  
Zhou, Jianying 1, 376  
Zhou, Yongbin 14  
Zhu, Huafei 207, 389