

Sihan Qing
Chris J. Mitchell
Guilin Wang (Eds.)

LNCS 5927

Information and Communications Security

11th International Conference, ICICS 2009
Beijing, China, December 2009
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

Alfred Kobsa

University of California, Irvine, CA, 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

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Sihan Qing Chris J. Mitchell
Guilin Wang (Eds.)

Information and Communications Security

11th International Conference, ICICS 2009
Beijing, China, December 14-17, 2009
Proceedings

Volume Editors

Sihan Qing
Chinese Academy of Sciences, Institute of Software
Beijing 100080, China
E-mail: qsihan@mail.ss.pku.edu.cn

Chris J. Mitchell
University of London, Information Security Group
Egham, Surrey TW20 0EX, United Kingdom
E-mail: c.mitchell@rhul.ac.uk

Guilin Wang
University of Birmingham, School of Computer Science
Birmingham, B15 2TT, United Kingdom
E-mail: g.wang@cs.bham.ac.uk

Library of Congress Control Number: 2009940402

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-642-11144-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-11144-0 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.com

© Springer-Verlag Berlin Heidelberg 2009
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12809325 06/3180 5 4 3 2 1 0

Preface

The 11th International Conference on Information and Communications Security (ICICS 2009) was held in Beijing, China during December 14–17, 2009. The ICICS conference series is an established forum that brings together people from universities, research institutes, industry and government institutions, who work in a range of fields within information and communications security. The ICICS conferences give attendees the opportunity to exchange new ideas and investigate developments in the state of the art. In previous years, ICICS has taken place in the UK (2008), China (2007, 2005, 2003, 2001 and 1997), USA (2006), Spain (2004), Singapore (2002), and Australia (1999). On each occasion, as on this one, the proceedings have been published in the Springer LNCS series.

In total, 162 manuscripts from 20 countries and districts were submitted to ICICS 2009, and a total of 37 (31 regular papers plus 6 short papers) from 13 countries and districts were accepted (an acceptance rate of 23%). The accepted papers cover a wide range of disciplines within information security and applied cryptography. Each submission to ICICS 2009 was anonymously reviewed by three or four reviewers. We are very grateful to members of the Program Committee, which was composed of 44 members from 14 countries; we would like to thank them, as well as all the external referees, for their time and their valuable contributions to the tough and time-consuming reviewing process.

In addition to the contributed speakers, the program also featured two invited speakers in the technical track. We are grateful to Richard A. Kemmerer (University of California, Santa Barbara, USA), and Wenbo Mao (EMC, USA), for accepting our invitation to speak. We also thank the keynote speakers from TCG on the first day of the conference, which was devoted to the industrial aspects of trusted and trustworthy computing.

ICICS 2009 was organized and hosted by the Institute of Software, Chinese Academy of Sciences (CAS), and the Institute of Software and Microelectronics, Peking University in co-operation with International Communications and Information Security Association (ICISA). The conference was sponsored by the National Natural Science Foundation of China under Grant No. 60573042 and No. 60970135, the Microsoft Corporation, Beijing Tip Technology Corporation, and the Trusted Computing Group (TCG).

We would like to thank Guilin Wang for his great work in arranging the publishing of the proceedings, and Dongmei Liu for her great contribution to the pre-conference arrangements and helping with many local details. Finally, we would like to thank the authors who submitted their papers to ICICS 2009, and the attendees from all around the world.

October 2009

Sihan Qing
Chris J. Mitchell

ICICS 2009

11th International Conference
on Information and Communications Security

Beijing, China
December 14–17, 2009

Organized by

Institute of Software, Chinese Academy of Sciences (CAS)
Institute of Software and Microelectronics, Peking University, China

Sponsored by

National Natural Science Foundation of China (NNSFC)
The Microsoft Corporation
Beijing Tip Technology Corporation, China
Trusted Computing Group (TCG)

In co-operation with

International Communications and Information Security Association (ICISA)

General Chair

Fuqing Yang Peking University, China

Vice Chair

Zhong Chen Peking University, China

Program Chairs

Sihan Qing Chinese Academy of Sciences and Peking University,
China

Chris J. Mitchell Royal Holloway, University of London, UK

Program Committee

Mikhail Atallah Purdue University, USA
Tuomas Aura Microsoft Research, UK
Thomas Berson Anagram Laboratories, USA
Alex Biryukov University of Luxembourg, Luxembourg

Srdjan Capkun	ETH Zurich, Switzerland
Chin-Chen Chang	Feng Chia University, Taiwan
Liqun Chen	Hewlett-Packard Laboratories, Bristol, UK
Zhong Chen	Peking University, China
Bruno Crispo	University of Trento, Italy
Edward Dawson	Queensland University of Technology, Australia
Dengguo Feng	Chinese Academy of Sciences, China
Yong Guan	Iowa State University, USA
Yeping He	Chinese Academy of Sciences, China
James Heather	University of Surrey, UK
Chi-Sung Laih	National Cheng Kung University, Taiwan
Gaicheng Li	Peking University, China
Yingjiu Li	Singapore Management University, Singapore
Javier Lopez	University of Malaga, Spain
Wenbo Mao	EMC Research, China
Peng Ning	North Carolina State University, USA
Xinxin Niu	Beijing University of Posts and Telecommunications, China
Eiji Okamoto	University of Tsukuba, Japan
Jean-Jacques Quisquater	UCL Crypto Group, Belgium
Kai Rannenber	Goethe University Frankfurt, Germany
Indrajit Ray	University of Birmingham, UK
Bimal Roy	Indian Statistical Institute, India
Mark Ryan	University of Birmingham, UK
Kouichi Sakurai	Kyushu University, Japan
Qingni Shen	Peking University, China
Miguel Soriano	Technical University of Catalonia, Spain
Jinshu Su	National University of Defense Technology, China
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Future University Hakodate, Japan
Guilin Wang	University of Birmingham, UK
Weiping Wen	Peking University, China
Andreas Wespi	IBM Zurich Research Laboratory, Switzerland
Duncan S. Wong	City University of Hong Kong, China
Wenling Wu	Chinese Academy of Sciences, China
Yongdong Wu	Institute for Infocomm Research, Singapore
Yixian Yang	Beijing University of Posts and Telecommunications, China
Alec Yasinsac	Florida State University, USA
Wentao Zhang	Chinese Academy of Sciences, China
Jianying Zhou	Institute for Infocomm Research, Singapore
Qiming Zhou	Chinese Academy of Sciences, China

Publication Chair

Guilin Wang	University of Birmingham, UK
-------------	------------------------------

Organizing Committee

Qing Yu	Beijing Tip Technology Corporation, China (Chair)
Dongmei Liu	Chinese Academy of Sciences, China
Qiming Zhou	Chinese Academy of Sciences, China
Qingni Shen	Peking University, China
Weiping Wen	Peking University, China
Gaicheng Li	Peking University, China

External Reviewers

Man Ho Au	Ghassan Karame	Falk Wagner
Ahmed Azab	Wen-Chung Kuo	Licheng Wang
Goekhan Bal	Stewart Kowalski	Shaobing Wang
Bruno Blanchet	Fagen Li	Shengyuan Wang
Jijun Cao	Qiming Li	Xiaofeng Wang
Pei-Te Chen	Rongsheng Li	Christian Weber
Qingkui Chen	Chuanguou Lin	Ralf-Philipp Weinmann
Xiaofeng Chen	Huan-Ping Liu	Wei Wu
Cheng-Kang Chu	Joseph K. Liu	Zhe Xia
Mauro Conti	Stephan Neuhaus	Jing Xu
Boris Danev	Ivica Nikolic	Ziyao Xu
Oscar Esparza	Federica Paci	Yanjiang Yang
Xiutao Feng	Thomas Plantard	Ziye Yang
Lothar Fritsch	Christina Poepper	Attila Altay Yavuz
Ge Fu	Mike Radmacher	Davide Zanetti
D. J. Guan	Kasper Rasmussen	Yingzhi Zeng
Juan Hernández-Serrano	Chun Ruan	Bin Zhang
Thomas S.	Steve Schneider	Fengli Zhang
Heydt-Benjamin	Masaaki Shirase	Jingcheng Zhang
Xinyi Huang	Mario Strasser	Mingwu Zhang
Qingguang Ji	Nils Tippenhauer	Baokang Zhao
Chunfu Jia	Joan Tomàs-Buliart	Wen Tao Zhu
Jianchun Jiang	Markus Tschersich	

Table of Contents

Invited Talks

How to Steal a Botnet and What Can Happen When You Do.....	1
<i>Richard A. Kemmerer</i>	
A User-Mode-Kernel-Mode Co-operative Architecture for Trustable Computing.....	2
<i>Wenbo Mao</i>	

Cryptanalysis

Security Evaluation of a DPA-Resistant S-Box Based on the Fourier Transform	3
<i>Yang Li, Kazuo Sakiyama, Shinichi Kawamura, Yuichi Komano, and Kazuo Ohta</i>	
Security Analysis of the GF-NLFSR Structure and Four-Cell Block Cipher.....	17
<i>Wenling Wu, Lei Zhang, Liting Zhang, and Wentao Zhang</i>	

Algorithms and Implementations

The RAKAPOSHI Stream Cipher	32
<i>Carlos Cid, Shinsaku Kiyomoto, and Jun Kurihara</i>	
Design of Reliable and Secure Multipliers by Multilinear Arithmetic Codes	47
<i>Zhen Wang, Mark Karpovsky, Berk Sunar, and Ajay Joshi</i>	
Hardware/Software Co-design of Public-Key Cryptography for SSL Protocol Execution in Embedded Systems.....	63
<i>Manuel Koschuch, Johann Großschädl, Dan Page, Philipp Grabher, Matthias Hudler, and Michael Krüger</i>	

Public Key Cryptography

Online/Offline Ring Signature Scheme	80
<i>Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou</i>	
Policy-Controlled Signatures.....	91
<i>Pairat Thorncharoensri, Willy Susilo, and Yi Mu</i>	

Public Key Encryption without Random Oracle Made Truly
 Practical 107
Puwen Wei, Xiaoyun Wang, and Yuliang Zheng

A Public-Key Traitor Tracing Scheme with an Optimal Transmission
 Rate 121
Yi-Ruei Chen and Wen-Guey Tzeng

Security Applications

Computationally Secure Hierarchical Self-healing Key Distribution for
 Heterogeneous Wireless Sensor Networks 135
Yanjiang Yang, Jianying Zhou, Robert H. Deng, and Feng Bao

Enabling Secure Secret Updating for Unidirectional Key Distribution
 in RFID-Enabled Supply Chains 150
*Shaoying Cai, Tieyan Li, Changshe Ma, Yingjiu Li, and
 Robert H. Deng*

Biometric-Based Non-transferable Anonymous Credentials 165
Marina Blanton and William M.P. Hudelson

Software Security

Secure Remote Execution of Sequential Computations 181
Ghassan O. Karame, Mario Strasser, and Srdjan Čapkun

Architecture- and OS-Independent Binary-Level Dynamic Test
 Generation 198
Gen Li, Kai Lu, Ying Zhang, Xicheng Lu, and Wei Zhang

System Security

Measuring Information Flow in Reactive Processes 211
Chunyan Mu

Trusted Isolation Environment: An Attestation Architecture with
 Usage Control Model 226
*Anbang Ruan, Qingni Shen, Liang Gu, Li Wang, Lei Shi,
 Yahui Yang, and Zhong Chen*

Denial-of-Service Attacks on Host-Based Generic Unpackers 241
Limin Liu, Jiang Ming, Zhi Wang, Debin Gao, and Chunfu Jia

Network Security

Predictive Pattern Matching for Scalable Network Intrusion
 Detection 254
Lucas Vespa, Mini Mathew, and Ning Weng

Deterministic Finite Automata Characterization for Memory-Based Pattern Matching	268
<i>Lucas Vespa and Ning Weng</i>	
A LoSS Based On-line Detection of Abnormal Traffic Using Dynamic Detection Threshold	283
<i>Zhengmin Xia, Songnian Lu, Jianhua Li, and Aixin Zhang</i>	
User-Assisted Host-Based Detection of Outbound Malware Traffic	293
<i>Huijun Xiong, Prateek Malhotra, Deian Stefan, Chehai Wu, and Danfeng Yao</i>	
Assessing Security Risk to a Network Using a Statistical Model of Attacker Community Competence	308
<i>Tomas Olsson</i>	

Short Papers I

Using the (Open) Solaris Service Management Facility as a Building Block for System Security	325
<i>Christoph Schuba</i>	
IntFinder: Automatically Detecting Integer Bugs in x86 Binary Program	336
<i>Ping Chen, Hao Han, Yi Wang, Xiaobin Shen, Xinchun Yin, Bing Mao, and Li Xie</i>	
A Comparative Study of Privacy Mechanisms and a Novel Privacy Mechanism	346
<i>Gunmeet Singh and Sarbjeet Singh</i>	

Database Security

Collusion-Resistant Protocol for Privacy-Preserving Distributed Association Rules Mining	359
<i>Xin-Jing Ge and Jian-Ming Zhu</i>	
GUC-Secure Join Operator in Distributed Relational Database	370
<i>Yuan Tian and Hao Zhang</i>	

Trust Management

TSM-Trust: A Time-Cognition Based Computational Model for Trust Dynamics	385
<i>Guangquan Xu, Zhiyong Feng, Xiaohong Li, Hutong Wu, Yongxin Yu, Shizhan Chen, and Guozheng Rao</i>	

Bring Efficient Connotation Expressible Policies to Trust Management 396
Yan Zhang, Zhengde Zhai, and Dengguo Feng

A User Trust-Based Collaborative Filtering Recommendation Algorithm..... 411
Fuzhi Zhang, Long Bai, and Feng Gao

Applied Cryptography

Fingerprinting Attack on the Tor Anonymity System 425
Yi Shi and Kanta Matsuura

Proactive Verifiable Linear Integer Secret Sharing Scheme 439
Chuangui Ma and Xiaofei Ding

A Multi-stage Secret Sharing Scheme Using All-or-Nothing Transform Approach 449
Mitra Fatemi, Taraneh Eghlidos, and Mohammadreza Aref

Digital Audio Watermarking Technique Using Pseudo-Zernike Moments 459
Xiangyang Wang, Tianxiao Ma, and Panpan Niu

Short Papers II

An Image Sanitizing Scheme Using Digital Watermarking 474
Masatoshi Noguchi, Manabu Inuma, Rie Shigetomi, and Hideki Imai

Adaptive and Composable Oblivious Transfer Protocols..... 483
Huafei Zhu and Feng Bao

Discrete-Log-Based Additively Homomorphic Encryption and Secure WSN Data Aggregation..... 493
Licheng Wang, Lihua Wang, Yun Pan, Zonghua Zhang, and Yixian Yang

Author Index 503

How to Steal a Botnet and What Can Happen When You Do

Richard A. Kemmerer

University of California, Santa Barbara, USA
kemmer@cs.ucsb.edu

Abstract. Botnets, which are networks of malware-infected machines that are controlled by an adversary, are the root cause of a large number of security threats on the Internet. A particularly sophisticated and insidious type of bot is Torpig, which is a malware program that is designed to harvest sensitive information (such as bank account and credit card data) from its victims. In this talk, we report on our efforts to take control of the Torpig botnet for ten days. Over this period, we observed more than 180 thousand infections and recorded more than 70 GB of data that the bots collected.

While botnets have been hijacked before, the Torpig botnet exhibits certain properties that make the analysis of the data particularly interesting. First, it is possible (with reasonable accuracy) to identify unique bot infections and relate that number to the more than 1.2 million IP addresses that contacted our command and control server during the ten day period. This shows that botnet estimates that are based on IP addresses are likely to report inflated numbers. Second, the Torpig botnet is large, targets a variety of applications, and gathers a rich and diverse set of information from the infected victims. This allowed us to perform interesting data analysis that goes well beyond simply counting the number of stolen credit cards. In this talk we will discuss the analysis that we performed on the data collected and the lessons learned from the analysis, as well as from the process of obtaining (and losing) the botnet.

A User-Mode-Kernel-Mode Co-operative Architecture for Trustable Computing

Wenbo Mao

EMC Research China
Mao_Wenbo@emc.com

Abstract. The Trusted Computing Group's technology takes a load-time code measurement approach to compute platform security, in which a code in a more privileged layer of the software stack is supposed to be able to maintain the correctness for one in a less privileged layer. In this work we first report evidences that this load-time code measurement method is insufficient for maintaining the software execution correctness. We propose a user-mode-kernel-mode co-operative architecture for trustable computing in which a secure application in user mode works in co-operation with the privileged system management software in kernel mode. We argue for the necessity of co-operation between a secure application and the secure service code in kernel mode, and showcase the practicality of this method.

Security Evaluation of a DPA-Resistant S-Box Based on the Fourier Transform

Yang Li¹, Kazuo Sakiyama¹, Shinichi Kawamura², Yuichi Komano²,
and Kazuo Ohta¹

¹ The University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan
{liyang,saki,ota}@ice.uec.ac.jp

² Toshiba Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan
{shinichi2.kawamura,yuichi1.komano}@toshiba.co.jp

Abstract. At CHES 2006, Prouff *et al.* proposed a novel S-box calculation based on the discrete Fourier transform as a first-order DPA countermeasure. At CHES 2008, Coron *et al.* showed that the original countermeasure can be broken by first-order DPA due to a biased mask and they proposed an improved algorithm. This paper shows that there is still a flaw in the Coron's S-box algorithm with respect to a practical software implementation. We pre-process the power traces to separate them into two subgroups, each has a biased mask. For the separated power traces, we propose two post analysis methods to identify the key. One is based on CPA attack against one subgroup, and the other is utilizing the difference of means for two subgroups and a pattern matching. Finally, we compare these two attack methods and propose an algorithm level countermeasure to enhance the security of Coron's S-box.

Keywords: Side channel attacks, Masking, Fourier transform, S-box, Probability density function.

1 Introduction

Side Channel Attacks (SCAs) which expose secret information using side channel leakages gained from the physical implementations of cryptosystem was first introduced by Kocher in 1996 [7]. Since then, various SCAs based on different leakage sources or different techniques are proposed. Among them, Differential Power Analysis (DPA) has been demonstrated to be very powerful. On the other hand, many DPA countermeasures were proposed both at algorithm level [2,4,6] and at logic level [13,11]. The goal of algorithm level countermeasure is masking every sensitive variable to suppress the dependence between the side channel leakage and the secret information. While the logic level countermeasures trend to improve the hardware gates to diminish the DPA leakage at the source.

At CHES 2006, Discrete Fourier Transform (DFT) was introduced by Prouff *et al.* as a general technique achieving immunity against first order DPA [8]. The proposed AES S-box algorithm based on DFT is referred as *Prouff's S-box* in this paper. Two

years later, Coron *et al.* shows that Prouff’s S-box is vulnerable to first-order DPA due to a biased mask in their algorithm [3]. They also propose an improved version of S-box based on the DFT(referred as *Coron’s S-box* in this paper). In the Coron’s S-box, it is proved that all the intermediate variables are masked by random numbers and are uniformly distributed.

In this paper, we show that Coron’s S-box still has a flaw in terms of power analysis in a practical software implementation. In Coron’s S-box, according to the value of the input, the output of the S-box is calculated as a combination of pre-calculated parameters. A random mask bit is used switching this calculation into two power profiles randomly. Focusing on a practical software implementation of Coron’s S-box, we show that different values of this mask bit can be distinguished from the power consumption at certain points of the algorithm. Our pre-process analysis method can separate traces into two subgroups with biased mask, and it is similar to the power analysis using *Probability Density Function* (PDF) proposed by Schaumont and Tiri [12,10]. After grouping the power traces, two post-processing analysis methods are explained and compared. One is first order DPA attack against one subgroup, and the other one is calculating the Difference of Means (DoMs) for two subgroups and matching the peaks with pre-calculated patterns.

2 Background Information

2.1 Masking Countermeasures against First Order DPA

Both [8] and [3] explain that if all the *intermediate variables* during the calculation are independent of any *sensitive variable*, the implementation is immune to first order DPA. The intermediate variables refer to the variables manipulated during the calculation, and the sensitive variables can be calculated by a key guess and public variables (*i.e.* plaintext or ciphertext). In the masking countermeasures against first order DPA, it is necessary all the intermediate variables are masked by uniformly distributed random variables to be independent of the sensitive variables.

2.2 Coron’s S-Box Calculation Based on DFT

In order to explain the concept of Coron’s S-box briefly, we follow the notations used in [3]. The DFT function \widehat{F} for the original function F is defined with $Z=(Z_{n-1}, \dots, Z_0) \in \mathbb{F}_2^n$ by

$$\widehat{F}(Z) = \sum_{i \in \mathbb{F}_2^n} F(i)(-1)^{i \cdot Z}, \quad (1)$$

where $i = (i_{n-1}, \dots, i_0) \in \mathbb{F}_2^n$ and “ \cdot ” denotes the scalar product calculated by

$$i \cdot Z = \bigoplus_{j=0}^{n-1} i_j Z_j. \quad (2)$$

If we perform the DFT operation against \widehat{F} again, we have $\widehat{\widehat{F}} = 2^n F$ as

$$F(Z) = \frac{1}{2^n} \sum_{i \in \mathbb{F}_2^n} \widehat{F}(i)(-1)^{i \cdot Z}. \quad (3)$$

In order to mask all the intermediate variables in calculating Eq. (3), three n -bit random numbers R_1, R_3 and $R_4 \in \mathbb{F}_2^n$ and one-bit random number $R_2 \in \mathbb{F}_2$ are necessary. For the input $(\tilde{Z} = Z \oplus R_1, R_1)$, the 3-tuple output $((-1)^{R_2} F(Z) + R_3, R_2, R_3)$ is calculated by

$$(-1)^{R_2} F(Z) + R_3 \bmod 2^n = \frac{1}{2^n} \left(R' + \sum_{i \in \mathbb{F}_2^n} \hat{F}(i) (-1)^{R_2 \oplus (i \cdot \tilde{Z}) \oplus (i \cdot R_1)} \right), \quad (4)$$

where $R' = 2^n R_3 + R_4$.

We denote the scalar product function $X, Y \mapsto X \cdot Y$ by SP and the function $X, T \mapsto \hat{F}(X)(-1)^T$ by SFT. Then, the calculation used for Coron's S-box can be described as shown in Alg. 1. The operation \boxplus denotes addition modulo 2^{2n} .

Algorithm 1. Calculation Steps for Coron's S-box [3]

INPUTS: A masked value $\tilde{Z} = Z \oplus R_1$ and the mask R_1

OUTPUT: The 3-tuple output $((-1)^{R_2} F(Z) + R_3 \bmod 2^n, R_3, R_2)$

1. Generate a random bit R_2

2. Generate two n -bit random R_3 and R_4

3. $result \leftarrow 2^n R_3 + R_4$

4. **for** i **from** 0 **to** $2^n - 1$ **do**

5. $T_1 \leftarrow SP(i, \tilde{Z})$ $[T_1 = i \cdot \tilde{Z}]$

6. $T_1 \leftarrow T_1 \oplus R_2$ $[T_1 = R_2 \oplus i \cdot \tilde{Z}]$

7. $T_2 \leftarrow SP(i, R_1)$ $[T_2 = i \cdot R_1]$

8. $T_1 \leftarrow T_1 \oplus T_2$ $[T_1 = R_2 \oplus i \cdot Z]$

9. $T_1 \leftarrow SFT(i, T_1)$ $[T_1 = \hat{F}(i)(-1)^{R_2 \oplus i \cdot Z}]$

10. $result \leftarrow result \boxplus T_1$

11. **end** $[result = (2^n R_3 + R_4) \boxplus \sum_{i=0}^{2^n-1} \hat{F}(i)(-1)^{R_2 \oplus i \cdot Z}]$

12. $result = result \gg n$ $[result = ((-1)^{R_2} F(Z) + R_3) \bmod 2^n]$

13. **return** $(result, R_3, R_2)$

3 The Flaw of Coron's S-Box and Attack Principle

3.1 Target Steps in the Coron's S-Box

Coron *et al.*'s paper proved that all the intermediate variables manipulated in Alg. 1 are well masked. However, there still exist several target steps in Alg. 1 even first-order DPA does not work directly on Coron's S-box. We review several steps of Coron's S-box considering a practical software implementation as follows.

The operation for $\hat{F}(i)$, $i \in \{0, 1, \dots, N\}$, is decided by $(-1)^{R_2 \oplus i \cdot \tilde{Z} \oplus i \cdot R_1}$ in Eq. (4), which can be further transformed as

$$\begin{aligned} (-1)^{R_2 \oplus i \cdot \tilde{Z} \oplus i \cdot R_1} &= (-1)^{R_2 \oplus i \cdot (Z \oplus R_1) \oplus i \cdot R_1} \\ &= (-1)^{R_2 \oplus i \cdot Z \oplus i \cdot R_1 \oplus i \cdot R_1} \\ &= (-1)^{R_2} (-1)^{i \cdot Z}. \end{aligned} \quad (5)$$

We notice R_2 is a one-bit mask (*i.e.* $R_2 \in \{0, 1\}$) which affects steps 8, 9 and 10 of Alg. 1 in every loop, where steps 9 and 10 perform operations as

$$\text{result} \leftarrow \text{result} + \widehat{F}(i)(-1)^{R_2}(-1)^{i \cdot Z}. \quad (6)$$

In Eq. (6) the value of $(-1)^{R_2}(-1)^{i \cdot Z}$ at step 9 directly decides the operation (addition or subtraction) at step 10. Here, we assume that addition with $-\widehat{F}(0)$ is equivalent to subtraction with $\widehat{F}(0)$ in terms of power consumption. And we denote the addition and subtraction operations by *+operation* and *-operation*, respectively. As the *-operation* involves an additional complement transform compared with the *+operation* in a practical micro-processor, there may exist some difference between the *+* and *-* operations in power consumption. Here we denote steps 9 and 10 as the *target steps*.

By way of experiment, we measured the power consumption when executing an experimental C code that operates only steps 8, 9 and 10 using fixed values of $R_2 \oplus i \cdot Z$ and $\widehat{F}(0)$. As shown in Fig. 1, an obvious power difference was observed between the cases of the *+* and *-* operations that correspond to $R_2 \oplus i \cdot Z = 0$ and 1, respectively.

3.2 Experiment Setup and Overview of Our Attacks

We implemented Coron's S-box over a composite field $\mathbb{F}_{2^4}^2$ based on Alg. 1 on the SASEBO-G (Side-channel Attack Standard Evaluation Board, type-G) [9,5]. The SASEBO-G is designed to develop evaluation schemes against physical attacks with FPGAs. We use a 32-bit reconfigurable CPU called Microblaze on a Xilinx FPGA (Vertex-II pro, xc2vp30) for the experiment. Coron's S-box is written in C code and 1-MHz clock is used for executing the compiled code. The power consumption of the CPU is obtained by measuring the voltage drop of a resistor inserted between FPGA's GND pins and the ground of the board. As shown in Fig. 2, where the 8-bit plaintext, the 8-bit secret key and the unmasked input of the S-box are denoted as P , K and a , respectively. And Z is unmasked input of Alg. 1.

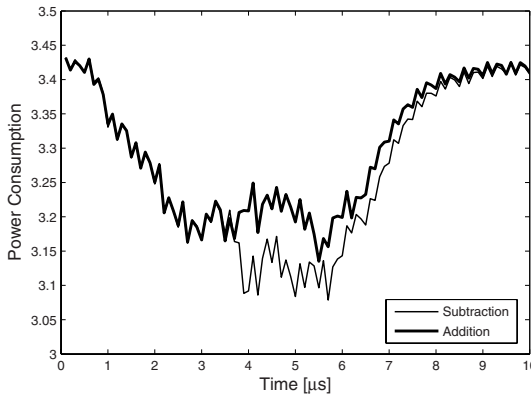


Fig. 1. Averaged power traces for steps 8, 9 and 10 using fixed values of $R_2 \oplus i \cdot Z$ and $\widehat{F}(0)$

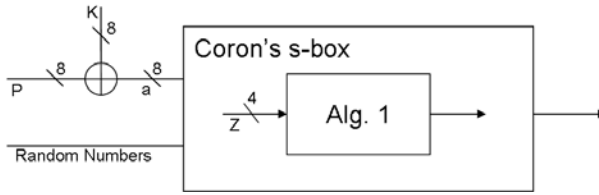


Fig. 2. The attacked S-box model

P is randomly chosen and all random variables are set to be uniformly distributed. The S-box calculation is repeated for 4000 times and the averaged power traces are shown in Fig. 3.

In Fig. 3, sixteen patterns can be clearly recognized corresponding to the 16-time loop of Alg. 1 and furthermore the beginning and the ending of every loop can be roughly guessed. Although the difference of means corresponding to $+$ and $-$ operations can be distinguished, but it only relates to the masked intermediate values, so first order DPA doesn't work. However the mask that masking target steps are used sixteen times in Coron's S-box, attackers can use multiple points of power trace to reveal the secret information.

In our attacks, the goal of our pre-processing for power traces is separating the power traces into two subgroups with regards to the biased values of R_2 . This separation can be achieved by looking into the probability density function of the certain segment of the power traces and details will be explained later. After separation, this paper proposes two post-processing methods to identify the secret key. One is perform traditional DPA to one subgroup. The other method is calculation of the Difference of Means (DoM) for two subgroups and match the peaks pattern and operation pattern to reveal the secret information.

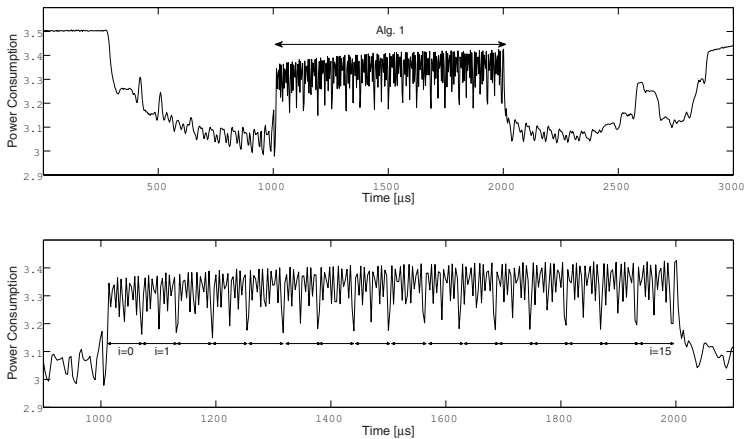


Fig. 3. The upper figure plots the averaged trace for a software implementation of Coron's S-box. The lower figure shows the magnified average traces focusing on the calculation of Alg. 1.

4 Pre-processing

The PDF scheme introduced by [12,10] involves using the histogram to specify the unbalanced influences caused by different values of a random mask bit. Then after a simple filtering and grouping of the power traces, every single subgroup should be vulnerable to traditional power analysis again. However it is normally difficult to specify the correct time when the target steps is performed. In the case of Coron’s S-box, we would like to see the PDF histogram at the moment of the + or – operation is performed. For the purpose of identifying the correct attacking point, we divide the end of the first loop into several parts along the time axis and plot the histogram for every part as shown in Fig. 4. That is, we propose a new power analysis called *PDF scanning method*, in which an attacker scans the power traces along the time axis and makes several plots for the PDF. In this way, we can seek out the exact position where + or – operation is performed by checking the shape of each plot for the PDF. In fact, as shown in Fig. 4, we could find a special PDF plot at the time part 8. The magnified figure is shown in Fig. 5.

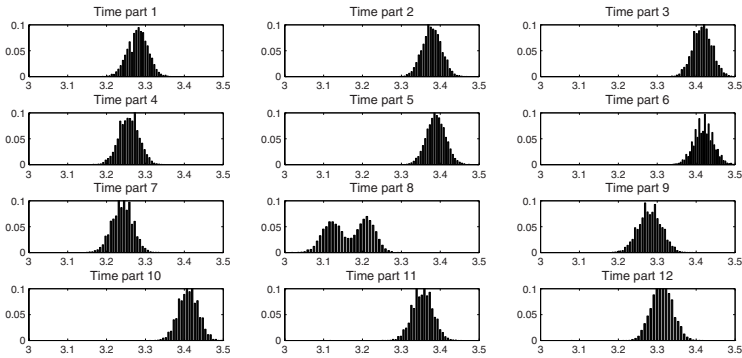


Fig. 4. Results of the PDF scanning method applying to around the end of the first loop of Alg. 1

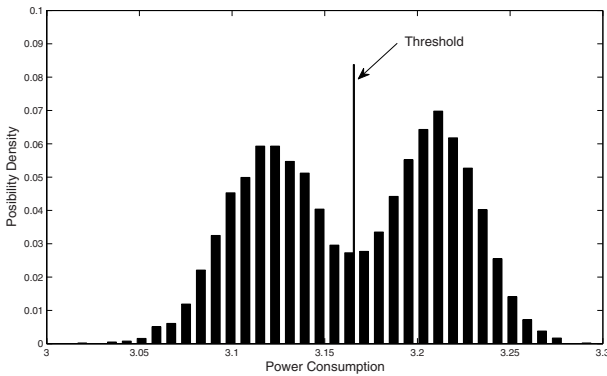


Fig. 5. PDF with two peaks

In Fig. 5, there are two peaks around the average power of 3.12 and 3.21. It is considered that two different distributions overlap each other and form those two peaks in the PDF figures as also discussed in [10]. By setting a threshold around the average power of 3.17, we separate the power trace into two subgroups. Notice that each subgroup still contains the power traces influenced by both the + and - operations because the threshold cannot separate two distributions perfectly. However, one subgroup contains the power traces where more + operations are executed than that the - operations, and vice versa for the other subgroup. Some discussion about the biased rate of R_2 in each subgroup is shown in Appendix A.

5 Two Post Analysis Methods

Our attack model used in this paper is similar to the one mentioned in [3] as

$$Z = \phi(P, K), \quad (7)$$

where ϕ is a Boolean function. The unmasked sensitive input of Alg. 1 Z is determined by value of plaintext P and the secret key K . Denote a key guess by K^* and the sensitive variable according to this K^* is denoted by Z_k .

5.1 CPA Attack against One Subgroup

Notice that when one of the inputs for the scale product operation is zero, the result becomes zero. Accordingly, when the first for-loop in Alg.1 is executed (*i.e.* $i = 0$ and $i \cdot Z = 0$), the value of $(-1)^{R_2 \oplus i \cdot Z} = (-1)^{R_2}$ is decided only by R_2 . When it is the first for-loop in Alg. 1 where the PDF separation is performed, even if P is randomly selected, the biased + operation or - operation in every subgroup directly corresponds to a biased R_2 . Therefore as a kind of post analysis, we can apply an improved version of traditional DPA called (CPA [1]), where the correlation coefficient is applied to determine the linear relationship between data. In the case of an implementation without masking countermeasures, the correct key guess produces a recognizable higher correlation coefficient between the vector of power traces and Z_k at certain moments. When the masking countermeasure is used but a mask is biased, we can still obtain similar results although the correlation coefficient will become lower.

For the case of our software implementation of Coron's S-box, we need two attacking points; one is a point for separating the power traces (denoted by *separation point*) and the other is for CPA attacks (denoted by *CPA point*). As mentioned previously we can find the separation point by using the PDF scanning method at the first loop ($i = 0$). Then, for the biased power traces, one of the remaining fifteen loops can be chosen as a candidate for the CPA point. Here, we choose the second loop ($i = 1$) as the CPA point.

When we used 10 000 power traces in separating groups by the PDF scanning method, the correct key can be distinguished from others with only about 200 traces. In other words, a good separation by the PDF scanning method leads to a small number of traces for CPA. However, the smaller the number of measurements become, the worse separation would be obtained. Therefore, we would anticipate that an optimal number of measurements exist between 200 and 10 000 in our experiment.

5.2 Calculation of DoMs for Two Subgroups and Pattern Matching

Before introducing another post analysis method, we review the Fourier transform used in Alg. 1 again. The basic formula of the S-box calculation based on the Fourier transform is described in Eq. (3). Denoting $2^n - 1$ by N , Eq. (3) can be expressed using the Hadamard matrix as

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \\ \vdots \\ F(N) \end{bmatrix} = \frac{1}{2^n} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots \\ 1 & -1 & 1 & -1 & \dots \\ 1 & 1 & -1 & -1 & \dots \\ 1 & -1 & -1 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ (-1)^{0 \cdot N} & (-1)^{1 \cdot N} & (-1)^{2 \cdot N} & (-1)^{3 \cdot N} & \dots \end{bmatrix} \begin{bmatrix} \widehat{F}(0) \\ \widehat{F}(1) \\ \widehat{F}(2) \\ \widehat{F}(3) \\ \vdots \\ \widehat{F}(N) \end{bmatrix}. \quad (8)$$

As the Alg. 1 is a 16-time loop, the 1-bit R_2 actually switches the practical calculation into two power profiles with regards to the + operation or - operation at sixteen positions. As long as the PDF separation at one position generates two subgroups with biased R_2 , the bias of + operation or - operation preserves for the rest fifteen target positions. So after calculating the Difference of Means for two subgroups, 16 peaks should appear. The important thing is that according to Eq. (2) the pattern of polarities of these peaks should directly correspond to the value of Z .

Recover the Unmasked Input of Coron’s S-box: As a new post analysis method, the input is fixed and 2000 power traces are used for the PDF separation. After that, we calculate the average traces from each subgroup, and the difference of them is calculated and plotted as shown in Fig. 6.

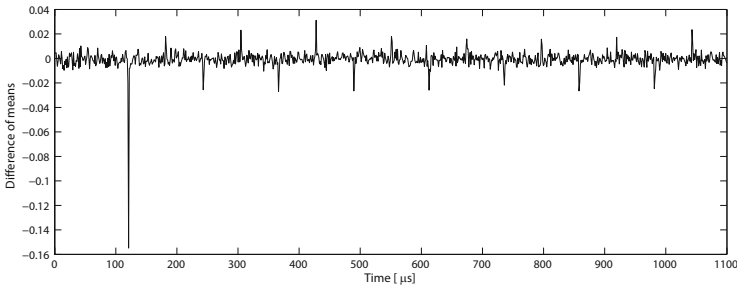


Fig. 6. Difference of mean traces for two groups ($Z = 1$)

As expected, Fig. 6 shows a trace with sixteen peaks with positive or negative directions. According to the Eq. (2), the pattern of 16 peaks starts from time 110 in Fig. 6 matches the operation pattern for $Z = 1$. Therefore the sensitive unmasked input of Alg. 1 Z is revealed by calculation of DoMs and the pattern marching.

Key Recovery from Unmasked Input: Real attackers can choose the first round of the AES as the target to identify the secret key. As shown in Fig. 2, we have $a = P \oplus K$.

As P is understandable and changeable by attackers, the recovery of a is equivalent to the recovery of K .

The unmasked input of S-box a is an 8-bit unknown variable, while the unmasked input of the Alg. 1 Z is a 4-bit variable. So given the value of Z , obviously the value of a cannot be identified. The relation between a and Z is determined according to the irreducible polynomial used in the combinational logic. By checking 256 possible values of a in our implementation, we obtain the relationship between Z and a as shown in Table 1.

From Table 1, we find that normally when Z is known, there are seventeen possible values of a . However there is a special one-to-one corresponding pair that in the case of $Z = 0$, $a = 0$. As a result, the 8-bit key can be successfully identified when the attacker obtain the DoM figure as shown in Fig. 7. Figure 7 with sixteen peaks at the

Table 1. The relationship between Z and possible unmasked input of S-box a in our implementation

Z	Possible unmasked input of S-box./ a (number of possible values).
0	0 (1)
1	1, 8, 29, 47, 53, 54, 57, 64, 74, 99, 102, 171, 179, 194, 211, 232, 239 (17)
2	9, 17, 44, 72, 76, 86, 92, 118, 123, 132, 134, 136, 157, 214, 233, 234, 245 (17)
3	3, 24, 35, 39, 42, 75, 90, 93, 95, 110, 113, 165, 170, 192, 206, 222, 230 (17)
4	25, 26, 28, 60, 62, 65, 87, 138, 142, 153, 164, 200, 208, 218, 224, 235, 251 (17)
5	5, 40, 49, 73, 91, 101, 105, 121, 126, 147, 178, 221, 225, 229, 231, 238, 244 (17)
6	4, 27, 32, 37, 51, 97, 116, 131, 141, 145, 151, 154, 188, 212, 216, 228, 250 (17)
7	2, 16, 58, 77, 94, 106, 108, 114, 125, 128, 148, 159, 189, 197, 198, 203, 204 (17)
8	10, 61, 80, 98, 127, 146, 161, 182, 199, 202, 209, 210, 213, 217, 242, 243, 252 (17)
9	6, 48, 70, 78, 79, 81, 84, 135, 150, 155, 167, 180, 186, 190, 215, 220, 226 (17)
10	12, 21, 45, 55, 85, 96, 103, 111, 115, 140, 156, 158, 162, 163, 168, 181, 223 (17)
11	13, 14, 30, 31, 69, 71, 82, 100, 104, 109, 112, 129, 166, 173, 193, 240, 248 (17)
12	22, 36, 38, 43, 46, 59, 66, 67, 68, 107, 117, 133, 137, 176, 195, 247, 249 (17)
13	20, 63, 89, 119, 122, 143, 149, 160, 169, 177, 185, 191, 196, 227, 253, 254, 255 (17)
14	11, 18, 19, 23, 33, 34, 88, 144, 152, 172, 183, 184, 201, 207, 236, 241, 246 (17)
15	7, 15, 41, 50, 52, 56, 83, 120, 124, 130, 139, 174, 175, 187, 205, 219, 237 (17)

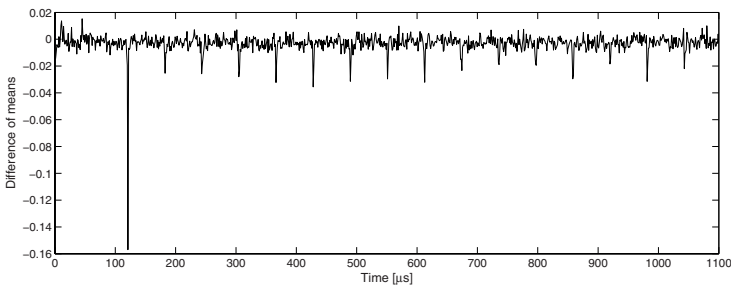


Fig. 7. Difference of mean traces for two groups ($Z = 0$)

same direction corresponds to $Z = 0$. Then according to the one-to-one correspondence in Table 1, $a = 0$. The corresponding 8-bit key piece is the bitwise inversion of the corresponding P . On the other hand, notice that when attackers get a pair of P_1 and P_2 corresponding two different groups of possible values of a , the attacker can restrain the key space by using the relationship as:

$$a_1 \oplus a_2 = (P_1 \oplus K) \oplus (P_1 \oplus K) = P_1 \oplus P_2. \quad (9)$$

We checked our implementation with Table 1, with only a pair of P_1 and P_2 that lead us to two groups of a , at most two key candidates are left. In other words, at most 3 different values of Z are needed to identify the key.

6 Comparison of Two Attack Methods

The common parts of those two attack methods (referred as PDF+CPA attack and PDF+DoMs attack) are the same target steps and the use of PDF separation. The major differences between them are listed as follows:

- In the PDF+CPA attack, after the PDF separation traditional CPA is operated (a comparison between PDF+CPA and Second-Order DPA is given in Appendix B), while in the PDF+DoMs attack, only DoMs is calculated and pattern matching is performed to reveal the secret information.
- In the PDF+CPA attack, for the subsequent CPA attack, P is randomly chosen, while in the PDF+DoMs attack, a fixed P is used 2000 times to identify the value of corresponding Z . PDF+DoMs attack needs fewer power traces, however attackers need more details of the implementation.
- In the PDF+CPA attack, the power consumption for the first two loops of Alg. 1 is enough for a successful attack, while in the PDF+DoMs attack, the power traces containing the entire 16 loops in Alg. 1 are necessary. Since we can have a better resolution for the power traces in PDF+CPA attack, we have a better PDF separation rate compared with that in the PDF+DoMs attack.
- Only the PDF at the first loop ($i = 0$ in Alg. 1) is meaningful in the PDF+CPA attack, while the PDF separation can be applied to any loop ($i = 0 \sim 15$ in Alg. 1) in the case of the PDF+DoMs attack.

7 Possible Countermeasures for the PDF+CPA Attack

We propose a possible countermeasure to enhance the security of Coron’s S-box. Considering that the loop index i is also a sensitive variable, we try to mask i with a random value as a countermeasure. For instance, we introduce a look-up table $q[l]$ ($0 \leq l \leq 15$) where each entry has a different integer randomly chosen from $0, 1, \dots, 15$ to randomize the loop index in the Coron’s S-box algorithm (see steps 2a and 4a in Alg. 2 in Appendix C). This countermeasure can prevent the PDF+CPA attack completely, however, in the case of $Z = 0$ the randomization of i cannot disorder the $+$ and $-$ operations. As a result, the peaks of DoMs still exists when $Z = 0$.

8 Conclusions and Future Work

This paper reviews the algorithm of the S-box calculation based on the DFT that is introduced as a first order DPA countermeasure [3]. Based on a practical software implementation, this paper presents a flaw of this s-box algorithm due to the difference of the power consumptions between different operations. By analyzing the power consumption related to this flaw, we use PDF of the power traces to separate them into two subgroups with biased operations. We propose and compare two post analysis methods in which a first-order DPA, and pattern matching after calculation of DoMs are used, respectively. Finally, we propose a possible algorithm-level countermeasure against the attack based on DPA attack. However, our countermeasure cannot prevent the attack based on DoMs and pattern matching completely, and the corresponding countermeasures should be considered in the future.

References

1. Brier, E., Clavier, C., Oliver, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
2. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
3. Coron, J.-S., Giraud, C., Prouff, E.: Attack and improvement of a secure S-box calculation based on the Fourier transform. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 1–14. Springer, Heidelberg (2008)
4. Coron, J.-S., Goubin, L.: On boolean and arithmetic masking against differential power analysis. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 231–237. Springer, Heidelberg (2000)
5. Research Center for Information Security (RCIS). Side-channel attack standard evaluation board (SASEBO), <http://www.rcis.aist.go.jp/special/SASEBO/index-en.html>
6. Itoh, K., Takenaka, M., Torii, N.: DPA countermeasure based on the masking method. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 440–456. Springer, Heidelberg (2002)
7. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
8. Prouff, E., Giraud, C., Aumônier, S.: Provably secure S-box implementation based on Fourier transform. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 216–230. Springer, Heidelberg (2006)
9. Rijmen, V.: Efficient implementation of the Rijndael S-box, citeseer.ist.psu.edu/293912.html
10. Schaumont, P., Tiri, K.: Masking and dual-rail logic don't add up. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 95–106. Springer, Heidelberg (2007)
11. Suzuki, D., Saeki, M., Ichikawa, T.: Random switching logic: A new countermeasure against DPA and second-order DPA at the logic level. IEICE Transaction on Fundamentals E90-A(1), 160–169 (2007)
12. Tiri, K., Schaumont, P.: Changing the odds against masked logic. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 134–146. Springer, Heidelberg (2007)

13. Tiri, K., Verbaauwhede, I.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: Proceedings of Design, Automation and Test in Europe Conference (DATE), pp. 246–251 (2004)

Appendix

A The Biased Rate of R_2 after PDF Separation

Suppose that the power consumption of + and – operations, denoted as P_{add} and P_{sub} respectively, both follow a normal distribution with different means of μ_{add} and μ_{sub} and the same deviation σ .

As long as there exists a comparably big difference between μ_{add} and μ_{sub} , We expect that the histogram of points at the appropriate time has the shape as shown in Fig. 8. The two peaks in Fig. 8 correspond to μ_{add} and μ_{sub} , respectively. The shape of

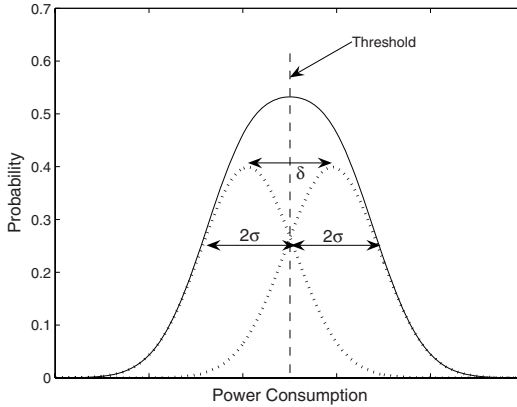


Fig. 8. PDF of two normal distributions

every mountain relates to the value of the standard deviation σ . Because $\mu_{add} \neq \mu_{sub}$, two distributions will not overlap perfectly. If we use the middle of the two distributions (*i.e.* the mean of all power traces) as a threshold to separate power traces into two subgroups, we can expect in this loop one subgroup contains the addition cases more than the subtraction cases, and vice versa for the other group. The biased operation is equivalent to the biased value of R_2 . Denoting the R_2 which accounts for more than a half in one group as *the major* R_2 , and *the separation rate* is defined as the ratio of the number of traces with the major R_2 to the total number of traces in one group. Denoting $\mu_{add} - \mu_{sub}$ and the separation rate by δ and α , respectively. The separation rate α can be computed by the ratio of δ to the standard deviation σ as shown in Fig. 9.

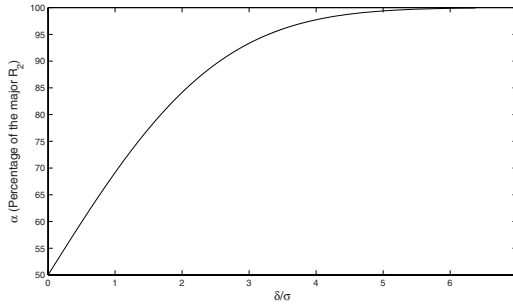


Fig. 9. The relationship between δ/σ and α

Suppose that the value of δ ($\delta > 0$) is specific in a certain implementation. The result of our attack scenario only relates to the value of σ . The smaller the σ is, the higher the separation rate α becomes and the bigger peak we can get¹.

B Comparison between PDF+CPA and Second Order DPA

If we stretch the interpretation of second-order DPA, PDF+CPA attack can be regarded as a kind of second-order DPA because two different attack points are used for power analysis. When permitting this extended interpretation, it is natural for us to have been successful in attacking Coron's S-box. However, the efficiency of PDF+CPA attack is better than or equivalent to first-order DPA in terms of the number of power traces. Therefore, the separation step in our two-step approach can be considered as a sort of the filtering processes that make it possible or easy to implement the succeeding first-order DPA.

It is worth mentioning that our attack can reduce the number of traces compared to conventional first-order DPA attacks. To explain this, we consider attacking Prouff's S-box (refer to [8] for the detailed algorithm). Coron *et al.* pointed out a flaw

$$T = i \cdot \tilde{Z} \oplus R_1 \cdot (\tilde{Z} \oplus i \oplus R_2) = i \cdot Z \oplus R_1 \cdot (\tilde{Z} \oplus R_2), \quad (10)$$

where R_1 and R_2 are n -bit random variables, i is the loop index and $\tilde{Z} = Z \oplus R_1$. Coron *et al.* utilized the fact that $R_1 \cdot (\tilde{Z} \oplus R_2)$ equals to 0 or 1, respectively with the probabilities of $17/32$ or $15/32$ (see the lemma in Sect. 4.1 in [3]). They applied first-order DPA by using a set of the power traces where the bias is not very prominent as the absolute difference between the probabilities λ is only $1/16$. And it turned out that λ in our experiment is much more than $1/16$ after the PDF separation for both cases of Coron's S-box and Prouff's S-box. This way, our attack can reduce the number of power traces compared to a straightforward first-order DPA.

¹ A rough approximation for the amplitude of the peak in difference of means is $(2\alpha - 1)\delta$.

C Algorithm of a Possible Countermeasure against PDF+CPA Attack

Algorithm 2. Coron's S-box with Randomized For Loop

INPUTS: A masked value $\tilde{Z} = Z \oplus R_1$ and the mask R_1

OUTPUT: The 3-tuple output $\left(((-1)^{R_2} F(Z) + R_3) \bmod 2^n, R_3, R_2 \right)$

1. Generate a random bit R_2
 2. Generate two n -bit random R_3 and R_4
 - 2a. Generate a look-up table $q[l]$ ($0 \leq l \leq 15$) (each entry has a different integer randomly chosen from $0, 1, \dots, 15$)
 3. $result \leftarrow 2^n R_3 + R_4$
 4. **for** l **from** 0 **to** $2^n - 1$ **do**
 - 4a. $i \leftarrow q[l]$
 - 5-12. The same as Alg. 1.
 13. **return** $(result, R_3, R_2)$
-

Security Analysis of the GF-NLFSR Structure and Four-Cell Block Cipher

Wenling Wu, Lei Zhang, Liting Zhang, and Wentao Zhang

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, P.R. China
w1@is.iscas.ac.cn

Abstract. The overall structure is one of the most important properties of block ciphers. At present, the most common structures include Feistel structure, SP structure, MISTY structure, L-M structure and Generalized Feistel structure. In [29], Choy et al. proposed a new structure called GF-NLFSR (Generalized Feistel-NonLinear Feedback Shift Register), and designed a new block cipher called Four-Cell which is based on the 4-cell GF-NLFSR. In this paper, we first study properties of the n -cell GF-NLFSR structure, and prove that for an n -cell GF-NLFSR, there exists an $(n^2 + n - 2)$ rounds impossible differential. Then we present an impossible differential attack on the full 25-round Four-Cell using this kind of 18-round impossible differential distinguisher together with differential cryptanalysis technique. The data complexity of our attack is $2^{111.5}$ and the time complexity is less than $2^{123.5}$ encryptions. In addition, we expect the attack to be more efficient when the relations between different round subkeys can be exploited by taking the key schedule algorithm into consideration.

Keywords: GF-NLFSR structure, Four-Cell block cipher, Impossible differential cryptanalysis, Data complexity, Time complexity.

1 Introduction

The overall structure is one of the most important properties of block ciphers, and it plays important roles in the round number choice, software and hardware implementation performances and so on. At present, the most often used structures include Feistel structure, SP structure, MISTY structure, L-M structure and Generalized Feistel structure. Feistel structure was introduced by H. Feistel in the design of Lucifer block cipher and later got famous since it was used in the design of DES. Feistel structure can transfer any function (usually called round function F) to a permutation. Now there are a lot of block ciphers employing the Feistel structure, such as Camellia, FEAL, GOST, LOKI, E2, Blowfish, RC5 and so on. The security of Feistel structure against differential and linear cryptanalysis was evaluated by many researchers, for example [1,2,3], and meanwhile there are many results such as [4,5,6,7,8,9,10] about the pseudorandomness of Feistel structure. Besides the Feistel structure, the other most often used structure is the SP structure, the well known block ciphers such as AES, Serpent

and ARIA all employ the SP structure. In each round of the SP structure, first a layer of key-dependent inversive function named S is applied to the input, and then applies a permutation or an inversive linear transformation named P . Hence the SP structure is very simple and clear, and S is usually called the confusion layer which achieves confusion in the cipher and P is usually referred to as the diffusion layer which diffuses efficiently. MISTY structure is another kind of important structures which was proposed by M. Matsui in [11], and it was used in the design of the block ciphers MISTY [12] and KASUMI [13]. There are many results about the security analysis of the MISTY structure such as in [14,15,16,17]. S. Vaudenay et al. named the structure of the block cipher IDEA as the L-M structure or Lai-Massey structure [18], and the FOX [19] cipher also employs a variant of the L-M structure. The generalized Feistel structure was first introduced by B. Schneier and J. Kelsey which can be considered as an unbalanced Feistel structure[20], and then many variants of generalized Feistel structure are proposed such as CAST-256-type [21], MARS-type [22], SMS4-type [23], CLEFIA-type [24] and so on. All these kinds of generalized Feistel structures have similar advantages such as decryption - encryption similarity and the inverse of round function is not necessary in decryption. Furthermore, this can make the design of round function more simple and flexible. The security analysis of generalized Feistel structure is very important when they are used to design new block ciphers, and there are many results [25,26,27,28] about the security of different kinds of generalized Feistel structures against the differential and linear cryptanalysis and also their pseudorandomness.

In [29], Choy et al. proposed a new structure called GF-NLFSR (Generalized Feistel-NonLinear Feedback Shift Register). It can be considered as an n -cell extension of combining the MISTY structure and Generalized Unbalanced Feistel Network together. The security of the structure against many attacks such as differential, linear, impossible differential and integral cryptanalysis are also considered in [29]. For an n -cell GF-NLFSR, an upper bound for the differential and linear hull probabilities for any $n + 1$ rounds are given, and a $2n - 1$ rounds impossible differential distinguisher and a $3n - 1$ rounds integral distinguisher on the n -cell GF-NLFSR are demonstrated. Furthermore, a new block cipher called Four-Cell which is based on the 4-cell GF-NLFSR was designed in [29]. The block and key size of Four-Cell are both 128-bit, and there are 25 rounds in total.

Impossible differential cryptanalysis [30] was first proposed by Biham, Biryukov and Shamir in 1999, and it was applied to analyze the Skipjack block cipher. Unlike differential cryptanalysis which exploits differentials with the highest possible probability, impossible differential cryptanalysis uses the differentials which hold with probability 0, which can thus be called impossible differential. The impossible differentials can usually be built in a miss-in-the-middle manner. Recently, impossible differential cryptanalysis had received worldwide attention, and its application to block ciphers such as AES, Camellia and MISTY all achieved very good results [31,32,33,34,35].

In [29], Proposition 3 stated that for an n -cell GF-NLFSR, there exist at most $2n - 1$ rounds impossible differential distinguishers using the U-method proposed

in [36]. However, we examine the property of n -cell GF-NLFSR structure and demonstrate that there exists a $(n^2 + n - 2)$ rounds impossible differential distinguisher. Then we present an impossible differential attack on the full 25-round Four-Cell using this kind of 18-round impossible differential distinguisher together with differential cryptanalysis technique.

This paper is organized as follows. In Section 2, we give a brief description of the n -cell GF-NLFSR structure and Four-Cell block cipher. In Section 3, we describe some useful properties of the n -cell GF-NLFSR structure and the $(n^2 + n - 2)$ rounds impossible differential. Then in Section 4, we present our impossible differential attack on the full 25-round Four-Cell block cipher. Finally, in Section 5 we summarize this paper.

2 The n -Cell GF-NLFSR Structure and Four-Cell Block Cipher

2.1 The n -Cell GF-NLFSR Structure

In this section, we will give a brief description of the n -cell GF-NLFSR structure, and Fig. 1 below illustrates one round of GF-NLFSR.

For an n -cell GF-NLFSR structure, suppose the size of the internal sub-block is m -bit, and then we can denote the mn -bit input block as $(x_1, x_2, x_3, \dots, x_n) \in (\{0, 1\}^m)^n$. If we denote the round subkey as sk , then the output of one round n -cell GF-NLFSR transformation is defined as follows.

$$\begin{aligned} x_2 &= x_2, \\ x_3 &= x_3, \\ &\dots \\ x_n &= x_n, \\ x_{n+1} &= f(x_1, sk) \oplus x_2 \oplus x_3 \dots \oplus x_n, \end{aligned}$$

where the output block is denoted as $(x_2, x_3, \dots, x_n, x_{n+1}) \in (\{0, 1\}^m)^n$. Note here the symbol \oplus is used to denote finite field addition (XOR) over $GF(2)^m$, and the function $f : \{0, 1\}^m \times \{0, 1\}^k \rightarrow \{0, 1\}^m$ is the round function. Specifically, for each fixed round key sk , the round function $f(\cdot, sk) : \{0, 1\}^m \rightarrow \{0, 1\}^m$ must be a permutation, or else the n -cell GF-NLFSR structure is not able to decrypt correctly. Therefore, in our later analysis, we will assume the round function f is a permutation when the round key is fixed.

2.2 Four-Cell Block Cipher

The block and key size of Four-Cell are both 128-bit, and it uses the 4-cell GF-NLFSR structure. Since the designers only give a rough suggestion for the key schedule algorithm, namely using a similar cipher with 26 rounds to generate the round keys needed. Hence in this paper, we will omit the key schedule and just assume that the round keys are randomly chosen. The encryption algorithm of Four-Cell can be described briefly as follows.

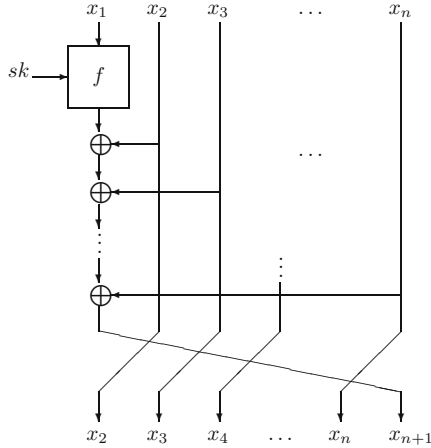


Fig. 1. One Round of n -Cell GF-NLFSR structure

Let the plaintext be denoted by $P = (x_1, x_2, x_3, x_4) \in (\{0, 1\}^{32})^4$, then after applying the full 25 rounds encryption, the 128-bit ciphertext can be denoted by C . Let $(x_i, x_{i+1}, x_{i+2}, x_{i+3}) \in (\{0, 1\}^{32})^4$ denote the input of the i -th round, then the output of the i -th round can be computed as follows.

$$\begin{aligned}
 x_{i+1} &= x_{i+1}, \\
 x_{i+2} &= x_{i+2}, \\
 x_{i+3} &= x_{i+3}, \\
 x_{i+4} &= f_i(x_i, sk_i) \oplus x_{i+1} \oplus x_{i+2} \oplus x_{i+3}.
 \end{aligned}$$

For rounds $i = 1, 2, \dots, 5$ and $i = 21, 22, \dots, 25$, the round keys are denoted as $sk_i \in \{0, 1\}^{32}$ and the round function is defined as $f_i(x_i, sk_i) = MDS(S(x_i \oplus sk_i))$. For rounds $i = 6, 7, \dots, 20$, the round keys are denoted as $sk_i = (sk_{i0}, sk_{i1}) \in (\{0, 1\}^{32})^2$, and the round function is defined as $f_i(x_i, sk_i) = S(MDS(S(x_i \oplus sk_{i0}))) \oplus sk_{i1}$.

Here in each round function, $S : (\{0, 1\}^8)^4 \rightarrow (\{0, 1\}^8)^4$ is four parallel 8×8 s -boxes, and the s -box is similar with the s -box used in the SubBytes operation in AES. The transformation $MDS : (\{0, 1\}^8)^4 \rightarrow (\{0, 1\}^8)^4$ is a 4-byte to 4-byte maximal distance separable transform with optimal branch number 5, and it is similar with the MixColumn operation in AES. In the end, the output after 25 rounds is XORed with a 128-bit post-whitening key $K_{26} = (k_{26}^1, k_{26}^2, k_{26}^3, k_{26}^4)$ to get the ciphertext, namely $C = (x_{26} \oplus k_{26}^1, x_{27} \oplus k_{26}^2, x_{28} \oplus k_{26}^3, x_{29} \oplus k_{26}^4)$.

3 Differential Property of the n -Cell GF-NLFSR Structure

For the n -cell GF-NLFSR structure, we can express the nm -bit input as n words which consists of m bits each. Suppose we have a pair of plaintexts

$X = (x_1, x_2, x_3, \dots, x_n)$ and $X^* = (x_1^*, x_2^*, x_3^*, \dots, x_n^*)$, and their difference is denoted by $\Delta X = (\Delta x_1, \Delta x_2, \dots, \Delta x_n)$, where $\Delta x_1 = x_1 \oplus x_1^*, \dots, \Delta x_n = x_n \oplus x_n^*$. Note the symbol 0 in the difference $\Delta X = (\Delta x_1, \Delta x_2, \Delta x_3, 0, \dots, 0)$ means that the corresponding byte difference is zero.

Lemma 1. *For the n -cell GF-NLFSR structure, there exists the following n rounds differential characteristic whose probability is equal to 1.*

$$(\Delta x_1, \Delta x_2, \dots, \Delta x_{i-1}, \Delta x_i, 0, \dots, 0) \xrightarrow{n \text{ rounds}} (\Delta y_1, \Delta y_2, \dots, \Delta y_i, \Delta y_{i+1}, 0, \dots, 0).$$

We denote this kind of differential characteristic as Δ_i , where $1 \leq i \leq n-1$, and these differential characteristics Δ_i satisfy the following two properties.

1. $\Delta y_1 \oplus \Delta y_2 \oplus \dots \oplus \Delta y_i \oplus \Delta y_{i+1} = 0$.
2. If $\Delta x_i \neq 0$, then $\Delta y_{i+1} \neq 0$.

Proof. Let the round function of Round i be $f_{sk_i}(x_i) = f_i(x_i, sk_i)$. Then when the round key sk_i is fixed, the round function f_{sk_i} must be a permutation, or else one can not decrypt correctly for the n -cell GF-NLFSR structure. According to the structure of the n -cell GF-NLFSR, we can get the following equations which are illustrated in Table 1.

$$\Delta y_1 = f_{sk_1}(x_1) \oplus f_{sk_1}(x_1 \oplus \Delta x_1) \oplus \Delta x_2 \oplus \dots \oplus \Delta x_i \quad (1)$$

$$\Delta y_i = f_{sk_i}(x_i) \oplus f_{sk_i}(x_i \oplus \Delta x_i) \oplus \Delta y_1 \oplus \dots \oplus \Delta y_{i-1} \quad (2)$$

$$\Delta y_{i+1} = \Delta y_1 \oplus \dots \oplus \Delta y_{i-1} \oplus \Delta y_i \quad (3)$$

Table 1. The n rounds differential characteristic of the n -cell GF-NLFSR structure

Round\Diff.	Δx_1	Δx_2	\dots	Δx_{i-1}	Δx_i	0	\dots	0
1	Δx_2	Δx_3	\dots	Δx_i	0	\dots	0	Δy_1
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$i-1$	Δx_i	0	\dots	0	0	Δy_1	\dots	Δy_{i-1}
i	0	\dots	0	0	Δy_1	\dots	Δy_{i-1}	Δy_i
$i+1$	0	\dots	0	Δy_1	\dots	Δy_{i-1}	Δy_i	Δy_{i+1}
$i+2$	0	\dots	Δy_1	\dots	Δy_{i-1}	Δy_i	Δy_{i+1}	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
n	Δy_1	Δy_2	\dots	Δy_i	Δy_{i+1}	0	\dots	0

According to Equ. (3), the following one round differential characteristic will hold with probability 1.

$$(0, \dots, 0, \Delta y_1, \dots, \Delta y_{i-1}, \Delta y_i, \Delta y_{i+1}) \xrightarrow{1 \text{ round}} (0, \dots, 0, \Delta y_1, \dots, \Delta y_i, \Delta y_{i+1}, 0).$$

Similarly, we can know that all the differential characteristics from Round $(i+2)$ to Round n in Table 1 hold with probability 1. Therefore, for the n -cell GF-NLFSR structure, there exists the following n rounds differential characteristic and its probability is equal to 1.

$$(\Delta x_1, \Delta x_2, \dots, \Delta x_{i-1}, \Delta x_i, 0, \dots, 0) \xrightarrow{n \text{ rounds}} (\Delta y_1, \Delta y_2, \dots, \Delta y_i, \Delta y_{i+1}, 0, \dots, 0)$$

Then according to Equ. (3), we can easily get the first property, i.e. $\Delta y_1 \oplus \Delta y_2 \oplus \dots \oplus \Delta y_i \oplus \Delta y_{i+1} = 0$. Therefore, in the following we only need to prove the second property.

According to Equ. (2) and Equ. (3), we can get the following equation.

$$\Delta y_{i+1} = \Delta y_1 \oplus \dots \oplus \Delta y_{i-1} \oplus \Delta y_i = f_{sk_i}(x_i) \oplus f_{sk_i}(x_i \oplus \Delta x_i).$$

When $\Delta x_i \neq 0$, we can conclude that $f_{sk_i}(x_i) \oplus f_{sk_i}(x_i \oplus \Delta x_i) \neq 0$ since the function f_{sk_i} is a permutation. Therefore, we get the second property, namely if $\Delta x_i \neq 0$, then $\Delta y_{i+1} \neq 0$. \square

Lemma 2. *For the inverse of the n -cell GF-NLFSR structure which is denoted as the n -cell GF-NLFSR⁻¹ structure, there exists the following $(2n - 2)$ rounds differential characteristic whose probability is equal to 1.*

$$(\beta, \beta, 0, \dots, 0) \xrightarrow{2n-2 \text{ rounds}} (?, \dots, ?, b_2, b_1, 0).$$

We denote this kind of differential characteristic as Δ_β^{-1} , where the symbol $?$ denotes an unknown difference and β, b_2, b_1 denote non-zero differences.

Proof. According to the structure of the n -cell GF-NLFSR⁻¹, we can get the following one round differential characteristic which holds with probability 1, and this kind of differential is illustrated in Table 2.

$$(\beta, \beta, 0, \dots, 0) \xrightarrow{1 \text{ round}} (0, \beta, \beta, 0, \dots, 0).$$

Similarly, all the differential characteristics from the Round 2 to Round $(n - 1)$ in Table 2 all hold with probability 1. Then in the Round n , if we denote the

Table 2. The $(2n - 2)$ rounds differential of the n -cell GF-NLFSR⁻¹ structure

Round \ Diff.	β	β	0	0	...	0
1	0	β	β	0	...	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$n - 2$	0	0	...	0	β	β
$n - 1$	0	0	...	0	0	β
n	b_1	0	0	...	0	0
$n + 1$	b_2	b_1	0	...	0	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$2n - 2$?	...	?	b_2	b_1	0

round function as g_n , then $b_1 = g_n(z) \oplus g_n(z \oplus \beta)$. Because the difference β is non-zero and the function g_n is a permutation, we can conclude that $b_1 \neq 0$. Similarly, in the $(n + 1)$ -th round we have $b_2 = g_{n+1}(w) \oplus g_{n+1}(w \oplus b_1)$, and thus we can conclude that $b_2 \neq 0$ since $b_1 \neq 0$.

Finally, according to the property of the n -cell GF-NLFSR⁻¹ structure, the differential characteristics from Round $(n + 2)$ to Round $(2n - 2)$ in Table 2 all hold with probability 1. Therefore, for the n -cell GF-NLFSR⁻¹ structure, there exists the following differential characteristic whose probability is equal to 1.

$$(\beta, \beta, 0, \dots, 0) \xrightarrow{2n-2 \text{ rounds}} (?, \dots, ?, b_2, b_1, 0). \quad \square$$

Theorem 1. *For the n -cell GF-NLFSR structure, there exists the following kind of $(n^2 + n - 2)$ rounds impossible differential where α and β are non-zero differences.*

$$(\alpha, 0, \dots, 0) \xrightarrow{n^2+n-2 \text{ rounds}} (\beta, \beta, 0, \dots, 0).$$

Proof. This kind of $(n^2 + n - 2)$ rounds impossible differential is constructed using the miss-in-the-middle technique. First we construct an $n(n - 1)$ rounds differential characteristic of the encryption direction and an $(2n - 2)$ rounds differential characteristic of the decryption direction whose probabilities are both equal to 1. Then if these two differential characteristics contradict each other in the middle, we get the $(n^2 + n - 2)$ rounds impossible differential. In Table 3 we illustrate this kind of impossible differential in detail.

When we choose the input difference as $(\alpha, 0, \dots, 0)$, we can construct an $n(n - 1)$ rounds differential with probability 1 as follows. First of all, based on Lemma 1, we can construct the following n rounds differential Δ_1 whose probability is equal to 1.

$$(\alpha, 0, \dots, 0) \xrightarrow{n \text{ rounds}} (\Delta x_1^2, \Delta x_2^2, 0, \dots, 0).$$

Since the input difference α is non-zero, according to property 1 and 2 of Lemma 1, we know that Δx_2^2 is also non-zero and $\Delta x_1^2 \oplus \Delta x_2^2 = 0$.

Then, we start with the input difference of $(\Delta x_1^2, \Delta x_2^2, 0, \dots, 0)$, and according to Lemma 1, we can construct again an n rounds differential Δ_2 whose probability is 1 as follows.

$$(\Delta x_1^2, \Delta x_2^2, 0, \dots, 0) \xrightarrow{n \text{ rounds}} (\Delta x_1^3, \Delta x_2^3, \Delta x_3^3, 0, \dots, 0).$$

Since Δx_2^2 is non-zero, we know that $\Delta x_3^3 \neq 0$ and $\Delta x_1^3 \oplus \Delta x_2^3 \oplus \Delta x_3^3 = 0$. Similarly, we can construct the i -th ($3 \leq i \leq n - 1$) n rounds differential Δ_i in turn. In the end, the $(n - 1)$ -th n rounds differential Δ_{n-1} is as follows, and we can conclude that $\Delta x_n^n \neq 0$ and $\Delta x_1^n \oplus \Delta x_2^n \oplus \Delta x_3^n \oplus \dots \oplus \Delta x_n^n = 0$.

$$(\Delta x_1^{n-1}, \Delta x_2^{n-1}, \dots, \Delta x_{n-1}^{n-1}, 0) \xrightarrow{n \text{ rounds}} (\Delta x_1^n, \Delta x_2^n, \Delta x_3^n, \dots, \Delta x_n^n).$$

By concatenating the above differentials together, we can get the following $n(n - 1)$ rounds differential whose probability is equal to 1 and Δx_n^n is non-zero.

$$(\alpha, 0, \dots, 0) \xrightarrow{n(n-1) \text{ rounds}} (\Delta x_1^n, \Delta x_2^n, \Delta x_3^n, \dots, \Delta x_n^n).$$

Table 3. The $(n^2 + n - 2)$ rounds impossible differential of n -cell GF-NLFSR structure

Round\Diff.	α	0	0	...	0	0
1	0	0	0	...	0	Δx_1^2
2	0	0	...	0	Δx_1^2	Δx_2^2
3	0	...	0	Δx_1^2	Δx_2^2	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
n	Δx_1^2	Δx_2^2	0	...	0	0
$n + 1$	Δx_2^2	0	0	...	0	Δx_1^3
$n + 2$	0	0	...	0	Δx_1^3	Δx_2^3
$n + 3$	0	...	0	Δx_1^3	Δx_2^3	Δx_3^3
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$2n$	Δx_1^3	Δx_2^3	Δx_3^3	0	...	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$n(n - 2)$	Δx_1^{n-1}	Δx_2^{n-1}	Δx_3^{n-1}	...	Δx_{n-1}^{n-1}	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$n(n - 1)$	Δx_1^n	Δx_2^n	Δx_3^n	...	Δx_{n-1}^n	Δx_n^n
	?	...	?	b_2	b_1	0
$n(n - 1) + 1$?	...	b_2	b_1	0	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$n^2 - 3$	b_2	b_1	0	0	...	0
$n^2 - 2$	b_1	0	0	...	0	0
$n^2 - 1$	0	0	...	0	0	β
n^2	0	0	...	0	β	β
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$n^2 + n - 3$	0	β	β	0	...	0
$n^2 + n - 2$	β	β	0	0	...	0

In the decryption direction, considering the inverse structure n -cell GF-NLFSR⁻¹, we get the following $2n - 2$ rounds differential characteristic with probability 1 according to Lemma 2.

$$(\beta, \beta, 0, \dots, 0) \xrightarrow{2n-2 \text{ rounds}} (?, \dots, ?, b_2, b_1, 0).$$

If we concatenate the above $n(n - 1)$ rounds differential of the encryption direction and the $(2n - 2)$ rounds differential of the decryption direction together, we can construct the following $(n^2 + n - 2)$ rounds impossible differential since they contradict each other at Δx_n^n .

$$(\alpha, 0, 0, \dots, 0) \xrightarrow{(n^2+n-2) \text{ rounds}} (\beta, \beta, 0, \dots, 0). \quad \square$$

4 Security Analysis of Four-Cell Block Cipher

According to Theorem 1, for Four-Cell block cipher which employs the 4-cell GF-NLFSR structure, there exists an 18 rounds impossible differential as follows.

$$(\alpha, 0, 0, 0) \xrightarrow{18 \text{ rounds}} (\beta, \beta, 0, 0)$$

By setting the 18-round impossible differential distinguisher in the middle rounds, we can present an impossible differential attack on the full 25-round Four-Cell by analyzing the first 4 rounds before and the last 3 rounds after the distinguisher. Note the round functions of the first 5 rounds and the last 5 rounds are all defined as $f_i(x_i, sk_i) = MDS(S(x_i \oplus sk_i))$.

Let the plaintext be $X = (x_1, x_2, x_3, x_4) \in (\{0, 1\}^{32})^4$, then the intermediate state after 3 rounds and 4 rounds encryption can be denoted as (x_4, x_5, x_6, x_7) and (x_5, x_6, x_7, x_8) respectively. Furthermore, the intermediate state after 22 rounds encryption can be denoted as $(x_{23}, x_{24}, x_{25}, x_{26})$ and the 128-bit ciphertext should be $C = (c_1, c_2, c_3, c_4) = (x_{26} \oplus k_{26}^1, x_{27} \oplus k_{26}^2, x_{28} \oplus k_{26}^3, x_{29} \oplus k_{26}^4)$. Suppose we choose another plaintext $X^* = (x_1^*, x_2^*, x_3^*, x_4^*) \in (\{0, 1\}^{32})^4$, and the plaintext difference can be denoted as $\Delta x_i = x_i \oplus x_i^*$.

Then for the last three rounds of Four-Cell, we have the following equations.

$$\begin{aligned} x_{27} &= MDS(S(x_{23} \oplus sk_{23})) \oplus x_{24} \oplus x_{25} \oplus x_{26}, \\ x_{28} &= MDS(S(x_{24} \oplus sk_{24})) \oplus x_{25} \oplus x_{26} \oplus x_{27}, \\ x_{29} &= MDS(S(x_{25} \oplus sk_{25})) \oplus x_{26} \oplus x_{27} \oplus x_{28}. \end{aligned}$$

If we denote $rk_{25} = k_{26}^1 \oplus k_{26}^2 \oplus k_{26}^3 \oplus k_{26}^4$, then the input of the Sbox layer for Round 25 can be computed as follows.

$$y_{25} = S^{-1}(MDS^{-1}(c_1 \oplus c_2 \oplus c_3 \oplus c_4 \oplus rk_{25})) = x_{25} \oplus sk_{25}.$$

Similarly, we can denote $rk_{24} = sk_{25} \oplus k_{26}^1 \oplus k_{26}^2 \oplus k_{26}^3$, and compute the input of the Sbox layer for Round 24 as follows.

$$y_{24} = S^{-1}(MDS^{-1}(c_1 \oplus c_2 \oplus c_3 \oplus y_{25} \oplus rk_{24})) = x_{24} \oplus sk_{24}.$$

Finally, for Round 23 the output of the round function is $c_1 \oplus c_2 \oplus y_{24} \oplus y_{25} \oplus sk_{25} \oplus sk_{24} \oplus k_{26}^1 \oplus k_{26}^2$. If we denote $rk_{23} = sk_{24} \oplus sk_{25} \oplus k_{26}^1 \oplus k_{26}^2$, then the input of the round function can be computed in a similar way.

$$y_{23} = S^{-1}(MDS^{-1}(c_1 \oplus c_2 \oplus y_{24} \oplus y_{25} \oplus rk_{23})) = x_{23} \oplus sk_{23}.$$

Therefore, considering that $\Delta x_{23} = \Delta y_{23}$, $\Delta x_{24} = \Delta y_{24}$, $\Delta x_{25} = \Delta y_{25}$ and $\Delta x_{26} = \Delta c_1$, we can obtain the values of $(\Delta x_{23}, \Delta x_{24}, \Delta x_{25}, \Delta x_{26})$ by just computing the values of y_{25} , y_{24} and y_{23} for a pair of ciphertexts $C = (c_1, c_2, c_3, c_4)$ and $C^* = (c_1^*, c_2^*, c_3^*, c_4^*)$.

For the first four rounds of Four-Cell, if we choose the plaintext difference as $(\Delta x_1, \Delta x_2, \Delta x_3, \Delta x_4) = (0, 0, 0, \alpha)$, then we can get the following equations.

$$\begin{aligned} \Delta x_5 &= \alpha, \\ \Delta x_6 &= 0, \\ \Delta x_7 &= 0, \\ \Delta x_8 &= MSD(S(x_4 \oplus sk_4)) \oplus MSD(S(x_4 \oplus \alpha \oplus sk_4)) \oplus \alpha. \end{aligned}$$

Here, $\Delta x_8 = 0$ holds if and only if $S(x_4 \oplus sk_4) \oplus S(x_4 \oplus \alpha \oplus sk_4) = MDS^{-1}(\alpha)$. Because the branch number of MDS is 5, there is at most one passive byte of α . For simplicity, we can assume the last byte of α is passive.

Let $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in (\{0, 1\}^8)^4$ and $\beta = (\beta_1, \beta_2, \beta_3, \beta_4) \in (\{0, 1\}^8)^4$, and then we will use the symbol $\alpha \xrightarrow{S} \beta$ to express that there exists $x_i = (x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}) \in (\{0, 1\}^8)^4$ such that $S(x_i) \oplus S(x_i \oplus \alpha) = \beta$.

Therefore, we can choose a set A which is defined as follows.

$$A = \{\alpha = (\alpha_1, \alpha_2, \alpha_3, 0) \in (\{0, 1\}^8)^4 \mid \alpha \xrightarrow{S} MDS^{-1}(\alpha)\}.$$

Note here the necessary condition for $\alpha \xrightarrow{S} MDS^{-1}(\alpha)$ is that α_1, α_2 , and α_3 should satisfy a linear relation (e.g. for the MDS used in AES, the linear relation is $0b \cdot \alpha_1 \oplus 0d \cdot \alpha_2 \oplus 09 \cdot \alpha_3 = 0$). Furthermore, for the Sbox of Four-Cell, the probability of $\alpha_i \xrightarrow{S} \beta_i$ holds is about 2^{-1} for $\forall \beta_i \in \{0, 1\}^8$. Therefore, the set A contains about $|A| \approx (2^8 - 1) \times (2^8 - 1) \times 2^{-1} \times 2^{-1} \times 2^{-1} \approx 2^{13}$ possible values. We also test this estimation using computer program, and with the same MDS and the Sbox used in AES, our searching result shows that the set A contains $7965 \approx 2^{12.96}$ possible values of α which is very close to the theory estimation.

After analyzing the first four rounds and the last three rounds of Four-Cell, we can set the 18-round impossible differential at Round 5 to Round 22 and apply an impossible differential attack on the full 25-round Four-Cell. The attack procedure consists of three steps, and we will utilize impossible differential attack technique together with some properties of the structure.

The first step of the attack is data collection. We first choose appropriate plaintext structures defined as follows.

$$S_P = \{(a_1, a_2, a_3, x_4)\},$$

where a_1, a_2, a_3 are 32-bit constants and the last byte of x_4 is also an 8-bit constants, namely $x_4 = (x_{4,1}, x_{4,2}, x_{4,3}, a_{4,4}) \in (\{0, 1\}^8)^4$, $x_{4,j} \in \{0, 1\}^8$. Therefore, each structure contains 2^{24} plaintexts and they can construct about $2^{24} \times 2^{13} / 2 \approx 2^{36}$ useful pairs whose plaintext differences satisfy the conditions listed above.

The second step of the attack is data filtering, in which we will discard all the useless pairs which do not satisfy the corresponding ciphertext difference. Note the output difference after the impossible differential distinguisher is $(\Delta x_{23}, \Delta x_{24}, \Delta x_{25}, \Delta x_{26}) = (\beta, \beta, 0, 0)$, and according to the structure of Four-Cell, the ciphertext difference $(\Delta c_1, \Delta c_2, \Delta c_3, \Delta c_4) = (\Delta x_{26}, \Delta x_{27}, \Delta x_{28}, \Delta x_{29})$ should satisfy the following two conditions.

$$\begin{aligned} \Delta c_1 &= 0, \\ \Delta c_1 \oplus \Delta c_2 \oplus \Delta c_3 \oplus \Delta c_4 &= 0. \end{aligned}$$

Therefore, the probability of a pair remains after this filtering is about 2^{-64} .

The third step of the attack is key recovery. First of all, for each guess of $(sk_{4,1}, sk_{4,2}, sk_{4,3})$ we can partially encrypt Round 4 to check if a pair satisfies the distinguisher. Note for each plaintext pair $X = (x_1, x_2, x_3, x_4)$ and $X^* = (x_1^*, x_2^*, x_3^*, x_4^*)$, a useful pair must satisfy that the output difference of the round

function in Round 4 equals to the input difference $x_4 \oplus x_4^*$. Therefore, based on this property we can discard some useless pairs to reduce the complexity in the following steps, and the probability of a pair remains after this filtering is about 2^{-21} . Then for all the remained pairs, we guess the values of rk_{25} and rk_{24} to decrypt Round 25 and Round 24 respectively. At last we recover the value of rk_{23} by differential techniques. Then we can discard all the wrong subkey guesses using the impossible differential sieving techniques.

In the following, we will describe the attack procedure in detail.

1. Data Collection: Choose 2^m structures and each structure is constructed as follows:

$$\begin{aligned} x_1 &= a_1, \\ x_2 &= a_2, \\ x_3 &= a_3, \\ x_4 &= (x_{4,1}, x_{4,2}, x_{4,3}, a_{4,4}), \end{aligned}$$

where (a_1, a_2, a_3) are 32-bit constants, $a_{4,4}$ is an 8-bit constant and the 3 bytes $(x_{4,1}, x_{4,2}, x_{4,3})$ take all the possible values of $(\{0, 1\}^8)^3$. Then each structure contains 2^{24} plaintexts, which can generate about $2^{24} \cdot 2^{13}/2 = 2^{36}$ plaintext pairs. Therefore, 2^m structures can generate about 2^{m+36} plaintext pairs.

2. Data Filtering: According to the property of ciphertext difference, for a useful pair the difference $(\Delta c_1, \Delta c_2, \Delta c_3, \Delta c_4)$ should satisfy the following conditions.

$$\begin{aligned} \Delta c_1 &= 0, \\ \Delta c_1 \oplus \Delta c_2 \oplus \Delta c_3 \oplus \Delta c_4 &= 0. \end{aligned}$$

Therefore, after this test the expected number of remaining pairs is about $2^{m+36} \cdot 2^{-64} = 2^{m-28}$.

3. For each guess of the 24-bit subkey $(sk_{4,1}, sk_{4,2}, sk_{4,3})$, proceed as follows:
 - (a) List all the possible values of rk_{23} as a table L .
 - (b) For each of the remaining plaintext pair $X = (x_1, x_2, x_3, x_4)$ and $X^* = (x_1^*, x_2^*, x_3^*, x_4^*)$, partially encrypt Round 4 to compute the following values respectively.

$$\begin{aligned} \gamma &= (s(x_{4,1} \oplus sk_{4,1}) \oplus s(x_{4,1}^* \oplus sk_{4,1}), \quad s(x_{4,2} \oplus sk_{4,2}) \oplus s(x_{4,2}^* \oplus sk_{4,2}), \\ &\quad s(x_{4,3} \oplus sk_{4,3}) \oplus s(x_{4,3}^* \oplus sk_{4,3}), \quad 0), \\ \lambda &= MDS^{-1}(x_4 \oplus x_4^*). \end{aligned}$$

Then check if $\gamma = \lambda$ holds, and if this is not the case, discard the corresponding plaintext pair. After this test, there remains about $2^{m-28} \cdot 2^{-21} = 2^{m-49}$ plaintext pairs.

- (c) Guess the value of $rk_{25} = k_{26}^1 \oplus k_{26}^2 \oplus k_{26}^3 \oplus k_{26}^4$, and for each of the remaining pair, whose ciphertexts are denoted as (c_1, c_2, c_3, c_4) and $(c_1^*, c_2^*, c_3^*, c_4^*)$ respectively, do as follows.
 - i. Compute the value of y_{25} as follows.

$$y_{25} = S^{-1}(MDS^{-1}(c_1 \oplus c_2 \oplus c_3 \oplus c_4 \oplus rk_{25})).$$

Note for the remaining pairs we have $\Delta y_{25} = 0$,
and $y_{25}^* = S^{-1}(MDS^{-1}(c_1^* \oplus c_2^* \oplus c_3^* \oplus c_4^* \oplus rk_{25})) = y_{25}$.

- ii. For each guess of the value $rk_{24} = sk_{25} \oplus k_{26}^1 \oplus k_{26}^2 \oplus k_{26}^3$, continue to compute the values of y_{24} and y_{24}^* as follows.

$$\begin{aligned} y_{24} &= S^{-1}(MDS^{-1}(c_1 \oplus c_2 \oplus c_3 \oplus y_{25} \oplus rk_{24})), \\ y_{24}^* &= S^{-1}(MDS^{-1}(c_1^* \oplus c_2^* \oplus c_3^* \oplus y_{25}^* \oplus rk_{24})). \end{aligned}$$

- iii. If we denote the decryption function of Round 23 as $g(z, rk_{23}) = S^{-1}(MDS^{-1}(z \oplus rk_{23}))$, then for each remaining pair the inputs of g are $c_1 \oplus c_2 \oplus y_{24} \oplus y_{25}$ and $c_1^* \oplus c_2^* \oplus y_{24}^* \oplus y_{25}^*$ respectively, and the output difference of g should be $y_{24} \oplus y_{24}^*$. Therefore, by making use of the difference distribution table of Sbox we can compute the corresponding value of subkey rk_{23} . Discard it from the table L .
- iv. If the table L is not empty after analyzing all the remaining pairs, we can output the value of rk_{23} remained in table L together with the corresponding guess of $(sk_{4,1}, sk_{4,2}, sk_{4,3})$, rk_{25} and rk_{24} as the correct subkey.

If we choose $m = 2^{87.5}$, then the number of useful pairs remained after the data filtering in Step 2 is about $2^{59.5}$. Hence there remains about $2^{38.5}$ pairs after the test of Step 3.b). In Step 3.c), according to the difference distribution table of Sbox, each pair can discard about one candidate of rk_{23} . Since there are 2^{32} possible values of rk_{23} in table L , then after analyzing all the $2^{38.5}$ remaining pairs, the probability of a subkey guess of rk_{23} still remains in L is about $(1 - 2^{-32})^{2^{38.5}} \approx e^{-2^{6.5}}$. Therefore, in Step 3.c.iv) the probability of a wrong subkey guess still remains after all the tests is about $2^{120} \times e^{-2^{6.5}} < 2^{-11}$, and this means that only the correct subkey will be output.

The data and time complexities of the attack can be estimated as follows. First of all, we choose $2^{87.5}$ structures which contains 2^{24} plaintexts each, and thus the data complexity of the attack is about $2^{24} \times 2^{87.5} = 2^{111.5}$ chosen plaintexts.

The time complexity of each step can be estimated roughly as follows. In Step 1, we need about $2^{111.5}$ encryptions. In Step 2 we have to check if the pair satisfies the ciphertext difference for all the $2^{123.5}$ pairs. Note the time needed for filtering is rather small which can be estimated as 2^{-3} -round encryption. Therefore, the time complexity of Step 2 is about $2^{123.5} \times \frac{1}{25} \times 2^{-3} < 2^{115.9}$ encryptions. In Step 3.b), we need to encrypt one round for each pair, which means that the time complexity is about $2^{24} \times 2^{59.5}/25 > 2^{78.9}$ encryptions. Similarly, the time complexities of Step 3.c.i) and Step 3.c.ii) are $2^{24} \times 2^{32} \times 2^{38.5} \times 1/25 < 2^{89.9}$ encryptions and $2^{24} \times 2^{32} \times 2^{32} \times 2^{38.5} \times 2/25 < 2^{122.9}$ encryptions respectively. In Step 3.c.iii), the operation to recover subkey rk_{23} from the difference distribution table of Sbox is rather simple and can be estimated as 1-round encryption. Then the time complexity of this step is about $2^{24} \times 2^{32} \times 2^{32} \times 2^{38.5} \times 1/25 < 2^{121.9}$ encryptions. Therefore, the total time complexity of the attack is less than $2^{123.5}$ encryptions.

5 Conclusion

In [29], Choy et al proposed a new structure called GF-NLFSR (Generalized Feistel-NonLinear Feedback Shift Register), and also examined the security of the structure against many attacks such as differential, linear, impossible differential and integral cryptanalysis. Furthermore, they designed a new block cipher called Four-Cell which is based on 4-cell GF-NLFSR structure. In this paper, we proved that for n -cell GF-NLFSR structure there exists $(n^2 + n - 2)$ rounds impossible differential. Then using this kind of 18-round impossible differential distinguisher together with some novel differential and impossible differential cryptanalysis techniques, we presented an impossible differential attack on the full 25-round Four-Cell. The data complexity of our attack is $2^{111.5}$ and the time complexity is less than $2^{123.5}$ encryptions. In addition, we expect the attack to be more efficient when the relations between different round subkeys can be exploited by taking the key scheduling algorithm into consideration.

Compared with the other kinds of generalized Feistel structures, the n -cell GF-NLFSR structure has some obvious advantage such as the ability of being parallel. However, if it is used to design a new block cipher, more work still need to be done about the security of the structure against various cryptanalysis and its pseudorandomness.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (No.60873259), and the National High-Tech Research and Development 863 Plan of China (No.2007AA01Z470). Moreover, the authors are very grateful to the anonymous referees for their comments and editorial suggestions.

References

1. Nyberg, K., Knudsen, L.: Provable Security against Differential Cryptanalysis. *Journal of Cryptology* 1(8), 156–168 (1995)
2. Knudsen, L.: Practically secure Feistel ciphers. In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809, pp. 211–221. Springer, Heidelberg (1994)
3. Kanda, M.: Practical security evaluation against differential and linear cryptanalyses for feistel ciphers with SPN round function. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, p. 324. Springer, Heidelberg (2001)
4. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing* 17(2), 373–386 (1988)
5. Lucks, S.: Faster Luby-Rackoff ciphers. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 189–203. Springer, Heidelberg (1996)
6. Patel, S., Ramzan, Z., Sundaram, G.: Towards making Luby-Rackoff ciphers optimal and practical. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 171–185. Springer, Heidelberg (1999)
7. Naor, M., Reingold, O.: On the construction of pseudorandom permutations Luby-Rackoff revisited. *Journal of Cryptology* 12(1), 9–66 (1999)
8. Maurer, U., Pietrzak, K.: The security of Many-Round Luby-Rackoff Pseudorandom Permutation. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 544–561. Springer, Heidelberg (2003)

9. Patarin, J.: Security of Random Feistel schemes with 5 or more rounds. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 106–122. Springer, Heidelberg (2004)
10. Wenling, W.: Pseudorandomness of Camellia-like scheme. *Journal of Computer Science and Technology* 12(1), 1–10 (2006)
11. Matsui, M.: New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 205–217. Springer, Heidelberg (1996)
12. Matsui, M.: New Block Encryption Algorithm MISTY. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 54–68. Springer, Heidelberg (1997)
13. ETSI, Universal Mobile Telecommunications System (UMTS), Specification of the 3GPP confidentiality and integrity algorithms, Document 2: Kasumi specification (2007), http://www.etsi.org/website/document/algorithms/ts_135202v070000p.pdf
14. Iwata, T., Yoshino, T., Yuasa, T., Kurosawa, K.: Round security and super-pseudorandomness of MISTY type structure. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 233–247. Springer, Heidelberg (2002)
15. Piret, G., Quisquater, J.-J.: Security of the MISTY Structure in the Luby-Rackoff Model: Improved Results. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 100–115. Springer, Heidelberg (2004)
16. Kang, J.S., Yi, O., Hong, D., et al.: Pseudorandomness of Misty-type Transformations and the Block Cipher KASUMI. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 60–73. Springer, Heidelberg (2001)
17. Iwata, T., Yagi, T., Kurosawa, K.: On the Pseudorandomness of KASUMI Type Permutations. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 217–289. Springer, Heidelberg (2003)
18. Vaudenay, S.: On the Lai-Massey Scheme. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 9–19. Springer, Heidelberg (1999)
19. Junod, P., Vaudenay, S.: FOX: a new Family of Block Ciphers. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 131–146. Springer, Heidelberg (2004)
20. Schneier, B., Kelsey, J.: Unbalanced Feistel Networks and Block Cipher Design. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 121–144. Springer, Heidelberg (2005)
21. Adams, C.: Constructing Symmetric Ciphers Using the CAST Design Procedure. *Designs, Codes and Cryptography* 12(3), 283–316 (1997)
22. MARS Block cipher, <http://www.nist.gov/aes/>
23. Specification of SMS4, Block Cipher for WLAN Products-SMS4 (in Chinese), <http://www.oscca.gov.cn/UpFile/200621016423197990.pdf>
24. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit block-cipher CLEFIA (Extended abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
25. Moriai, S., Vaudenay, S.: On the Pseudorandomness of Top-Level Schemes of Block Ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 289–302. Springer, Heidelberg (2000)
26. Nyberg, K.: Generalized Feistel networks. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 91–104. Springer, Heidelberg (1996)
27. Wu, W., Zhang, W., Lin, D.: On the Security of Generalized Feistel Scheme with SP Round Function. *International Journal Network Security* 2(3), 296–305 (2006)
28. Shirai, T., Shibutani, K.: On Feistel structures using a diffusion switching mechanism. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 41–56. Springer, Heidelberg (2006)

29. Choy, J., Chew, G., Khoo, K., Yap, H.: Cryptographic Properties and Application of a Generalized Unbalanced Feistel Network Structure. In: ACISP 2009. LNCS, vol. 5594, pp. 73–89. Springer, Heidelberg (2009)
30. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 12–23. Springer, Heidelberg (2003)
31. Phan, R.C.-W.: Impossible Differential Cryptanalysis of 7-round AES. *Information Processing Letters* 91(1), 33–38 (2004)
32. Zhang, W., Wu, W., Feng, D.: New Results on Impossible Differential Cryptanalysis of Reduced AES. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 239–250. Springer, Heidelberg (2007)
33. Wu, W., Zhang, W., Feng, D.: Impossible Differential Cryptanalysis of Reduced-Round ARIA and Camellia. *Journal of Computer Science and Technology* 22(3), 449–456 (2007)
34. Tsunoo, Y., Tsujihara, E., Shigeri, M., Saito, T., Suzaki, T., Kubo, H.: Impossible differential cryptanalysis of CLEFIA. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 398–411. Springer, Heidelberg (2008)
35. Dunkelman, O., Keller, N.: An Improved Impossible Differential Attack on MISTY1. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 441–454. Springer, Heidelberg (2008)
36. Kim, J.-S., Hong, S.H., Sung, J., Lee, S.-J., Lim, J.-I., Sung, S.H.: Impossible differential cryptanalysis for block cipher structures. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 82–96. Springer, Heidelberg (2003)

The RAKAPOSHI Stream Cipher

Carlos Cid¹, Shinsaku Kiyomoto², and Jun Kurihara²

¹ Information Security Group,
Royal Holloway, University of London
Egham, United Kingdom
`carlos.cid@rhul.ac.uk`

² KDDI R & D Laboratories Inc.
2-1-15 Ohara, Fujimino-shi, Saitama 356-8502, Japan
`kiyomoto@kddilabs.jp`, `kurihara@kddilabs.jp`

Abstract. In this paper, we introduce the RAKAPOSHI stream cipher. The algorithm is based on Dynamic Linear Feedback Shift Registers, with a simple and potentially scalable design, and is particularly suitable for hardware applications with restricted resources. The RAKAPOSHI stream cipher offers 128-bit security, and aims to complement the current eSTREAM portfolio of hardware-oriented stream ciphers.

1 Introduction

A stream cipher is a type of encryption algorithm that encrypts individual alphabet elements of plaintext, one at a time, with a time-varying transformation [28]. Stream ciphers are very popular due to their many attractive features: they are generally fast, can typically be efficiently implemented in hardware, have no (or limited) error propagation, and are particularly suitable for use in environments where no buffering is available and/or plaintext elements need to be processed individually. These are particularly important features in the telecommunication sector, and stream ciphers are ubiquitous in the field.

Recent years have witnessed an increase in the research of design and analysis of stream ciphers, primarily motivated by eSTREAM, the ECRYPT Stream Cipher Project [12]. eSTREAM was a multi-year project, which started in 2004, and had the objective of selecting a portfolio of promising stream cipher designs. The selection of algorithms was based on two usage profiles, corresponding to specific applications identified for stream ciphers of dedicated design:

- Profile 1: stream ciphers for software applications with high throughput.
- Profile 2: stream ciphers for hardware applications with highly restricted resources.

The project received 34 submissions, of which 16 were selected to the final phase [31]. The final portfolio was announced in April 2008, containing eight ciphers: four in profile 1 and four in profile 2 [4]. The portfolio was later revised [3],

due to new cryptanalytic results [15] against one of the selected ciphers in profile 2 (namely, the F-FSCR-H stream cipher [1]).

Despite the end of the eSTREAM project, the research area of analysis and design of stream ciphers remains active, with particularly eSTREAM portfolio ciphers continuing to attract much attention of the cryptographic community [8]. In this trend, we propose in this paper a new stream cipher, called RAKAPOSHI. The algorithm presents a simple and potentially scalable design, and is particularly suitable for hardware applications with restricted resources (and thus it would fall into profile 2 of the eSTREAM project). The motivation for such a proposal soon after the end of the eSTREAM project is manifold:

- The cipher provides 128-bit security. Most eSTREAM candidates in profile 2 employ 80-bit keys, as suggested in the original call for proposals [1]. However we believe 128-bit security is a very attractive feature when aiming for long-term deployment of such ciphers.
- The cipher uses a simple and elegant design, based on Dynamic Linear Feedback Shift Registers (Section 2). The design can be seen as a generalisation of constructions found in early designs [6, 33, 24], and may motivate a more detailed mathematical analysis of properties of such constructions.
- The cipher design and security evaluation incorporates lessons learned during the several years of extensive analysis in the eSTREAM process, and thus RAKAPOSHI is less likely to be susceptible to more recent attacks, such as initialisation attacks.
- More importantly, as noted by the eSTREAM selection committee, ciphers in the final portfolio are still very new, and analysis may not be mature enough to consider widespread deployment [3]. This was indeed illustrated by the need to revise the eSTREAM portfolio soon after its announcement. eSTREAM algorithms in profile 2 may in fact become the most popular stream ciphers in practice, due to the growing use of cryptographic mechanisms in small electronic devices (such as RFIDs). Thus we believe that alternatives to eSTREAM ciphers in this particular profile may prove to be desirable for future use.

In summary, we believe that RAKAPOSHI complements the current eSTREAM portfolio in profile 2, while presenting some attractive additional features, increasing thus the choice of secure lightweight stream ciphers suitable for hardware applications with restricted resources.

This paper is organised as follows. In Section 2 we give a brief overview of the main component of the RAKAPOSHI design (namely, DLFSR - Dynamic Linear Feedback Shift Registers). In Section 3 we describe the details of the cipher specification and design, and in Section 4 we describe the cipher operation. In Section 5 we present a provisional security evaluation of the cipher. In Section 6 we discuss some implementation aspects of the RAKAPOSHI cipher, and present our concluding remarks in Section 7.

¹ We note however that some designers later also defined 128-bit versions of the original 80-bit submissions to eSTREAM.

2 Dynamic Linear Feedback Shift Registers

A Dynamic Linear Feedback Shift Register (DLFSR) scheme is a general construction consisting usually of two registers: the first subregister \mathcal{A} , of length r , is clocked regularly and updated using a fixed mapping $\lambda_{\mathcal{A}}$. Subregister \mathcal{B} , of length n , is updated using a linear mapping $\lambda_{\mathcal{B}}^t$, which varies with time and depends of the state in register \mathcal{A} at time t . Thus subregister \mathcal{A} is used to select and dynamically modify the feedback function of LFSR \mathcal{B} , and as a result, \mathcal{B} presents an irregular updating mechanism, as opposed to the regular clocking that the register would present if a unique, fixed linear feedback function was used. In practice, the state of \mathcal{B} is used as input to a non-linear function to produce the output sequence of a stream cipher.

Although used in early ciphers, the general concept of Dynamic Linear Feedback Shift Registers seems to have been first proposed in open literature in the short article by Mita *et al* [29]. A simple example was presented, and properties of the output sequence were studied and compared with the output of a conventional LFSR of similar size. A more detailed characterisation of DLFSRs was presented in [27]. We note that we may generalise the idea to have non-linear updating functions, or possibly more than two subregisters.

Examples of stream ciphers based on the Dynamic Linear Feedback Shift Register primitive include the stop-and-go generator [6], LILI [33], dynamic feedback polynomial switch [21], and K2 [24]. The RAKAPOSHI stream cipher is in fact a successor of the K2 stream cipher, but aiming at low-cost hardware implementations. Furthermore, in the RAKAPOSHI design, the FSR \mathcal{A} uses a non-linear updating function, and the state of both subregisters are used as input to a non-linear output function to produce the cipher's keystream.

3 Cipher Design

In this section we describe the main design criteria for the RAKAPOSHI stream cipher, as well as the details of the cipher specification.

3.1 Design Criteria

The main criteria used in the design of RAKAPOSHI were: 128-bit security, use of elegant and structurally rich FSR-based construction, and competitive performance and hardware implementation requirements when compared to ciphers in profile 2 of the eSTREAM portfolio. Furthermore, the cipher design is such that increasing the speed of the algorithm may be efficiently done by implementation of circuits for parallelization of the algorithm.

The RAKAPOSHI parameters are the following:

Key length:	128 bits.
Initialisation Vector length:	192 bits.
NLFSR \mathcal{A} length r :	128 bits.
DLFSR \mathcal{B} length n :	192 bits.

Thus the size of the internal state $\mathcal{S} = (\mathcal{A}, \mathcal{B})$ is $s = r + n = 320$ bits. When aiming for 128-bit security, we note that the internal state size of the cipher should be at least 320 bits, in view of the TMTO attack recently proposed by Dunkelman and Keller [11] (see Section 5.1).

We define below the notation used in this paper:

User-provided Key: $\{k_0, k_1, \dots, k_{127}\}$
 Initialisation Vector (IV) : $\{iv_0, iv_1, \dots, iv_{191}\}$
 Registers of NLFSR at time t : $[a_t, a_{t+1}, \dots, a_{t+127}] = \mathcal{A}^t$
 Registers of DLFSR at time t : $[b_t, b_{t+1}, \dots, b_{t+191}] = \mathcal{B}^t$
 Keystream at time t : z_t

3.2 RAKAPOSHI Specification

The RAKAPOSHI stream cipher main component is the bit-oriented Dynamic Linear Feedback Shift Register (DLFSR). It consists of a 128-bit Non-Linear Feedback Shift Register and a 192-bit Linear Feedback Shift Register, denoted as registers \mathcal{A} and \mathcal{B} , respectively. The cipher uses two bits from the state of the NLFSR to select, and dynamically modify the (linear) feedback function of the LFSR. The cipher keystream is produced by combining the output of both registers with the output of a non-linear Boolean function over $(\mathbb{F}_2)^8$. This function takes as input six bits from the state of register \mathcal{B} and two bits from the state of register \mathcal{A} . The cipher schematic is depicted in Figure 1.

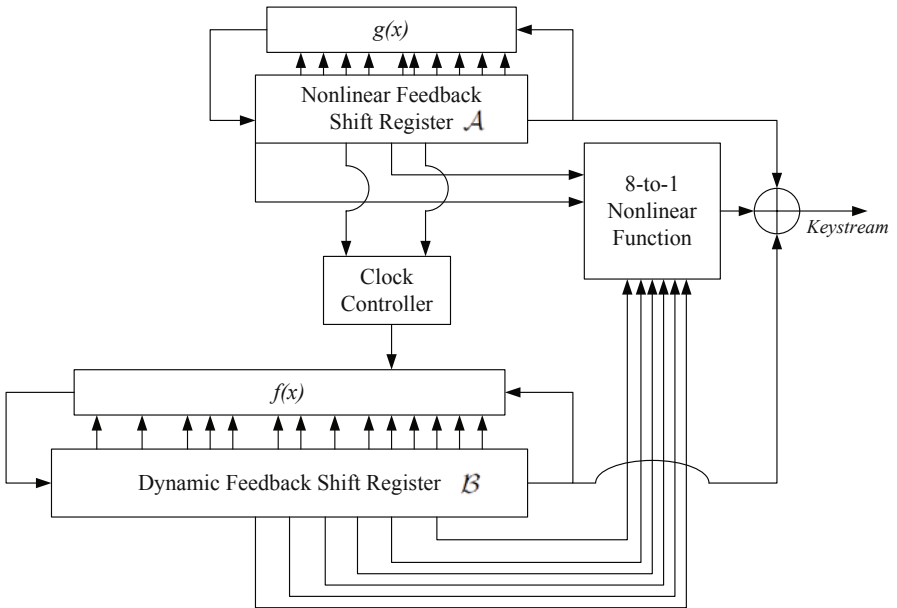


Fig. 1. RAKAPOSHI Stream Cipher

Non-Linear Feedback Shift Register \mathcal{A} . The cipher's register \mathcal{A} is a 128-bit NLSFR, defined using the feedback function

$$\begin{aligned} g(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = & x_1x_3x_9 + x_1x_7x_9 + x_5x_8 + x_2x_5 + \\ & x_3x_8 + x_2x_7 + x_9 + x_8 + x_7 + x_6 + \\ & x_5 + x_4 + x_3 + x_2 + x_1 + x_0 + 1, \end{aligned}$$

as $a_{t+128} = g(a_t, a_{t+6}, a_{t+7}, a_{t+11}, a_{t+16}, a_{t+28}, a_{t+36}, a_{t+45}, a_{t+55}, a_{t+62})$. Explicitly, we have the following recurrence relation:

$$\begin{aligned} a_{t+128} = & 1 \oplus a_t \oplus a_{t+6} \oplus a_{t+7} \oplus a_{t+11} \oplus a_{t+16} \oplus a_{t+28} \oplus a_{t+36} \oplus a_{t+45} \oplus \\ & a_{t+55} \oplus a_{t+62} \oplus a_{t+7}a_{t+45} \oplus a_{t+11}a_{t+55} \oplus a_{t+7}a_{t+28} \oplus \\ & a_{t+28}a_{t+55} \oplus a_{t+6}a_{t+45}a_{t+62} \oplus a_{t+6}a_{t+11}a_{t+62}. \end{aligned}$$

The feedback function g was selected according to criteria for non-linear feedback shift registers with maximum period [34] and consists of a primitive linear feedback shift register and a non-linear function of degree 3. This is a balanced function, of which the best linear approximation has bias 2^{-4} .

Linear Feedback Shift Register \mathcal{B} . The register \mathcal{B} is a 192-bit Dynamic LFSR, which can use four different linear recursive functions. These are selected using two bits from the state of register \mathcal{A} . Let c_0 and c_1 be the 42nd and 90th bits of register \mathcal{A} at time t , respectively (that is, $c_0 = a_{t+41}$ and $c_1 = a_{t+89}$). Then LFSR \mathcal{B} at time t is defined by the following characteristic polynomial:

$$\begin{aligned} f(x) = & x^{192} + x^{176} + c_0x^{158} + (1 + c_0)x^{155} + c_0c_1x^{136} + \\ & c_0(1 + c_1)x^{134} + c_1(1 + c_0)x^{120} + (1 + c_0)(1 + c_1)x^{107} + \\ & x^{93} + x^{51} + x^{49} + x^{41} + x^{37} + x^{14} + 1. \end{aligned}$$

We note that $f(x) \in \mathbb{F}_2[x]$ is a primitive polynomial for all four choices of (c_0, c_1) . Thus the recurrence relation for register \mathcal{B} is given by:

$$\begin{aligned} b_{t+192} = & b_t \oplus b_{t+14} \oplus b_{t+37} \oplus b_{t+41} \oplus b_{t+49} \oplus b_{t+51} \oplus b_{t+93} \oplus \\ & \overline{c_0} \cdot \overline{c_1} \cdot b_{t+107} \oplus \overline{c_0} \cdot c_1 \cdot b_{t+120} \oplus c_0 \cdot \overline{c_1} \cdot b_{t+134} \oplus c_0 \cdot c_1 \cdot b_{t+136} \oplus \\ & \overline{c_0} \cdot b_{t+155} \oplus c_0 \cdot b_{t+158} \oplus b_{t+176}, \end{aligned}$$

where $\overline{c_i} = 1 \oplus c_i$ represents the negation of c_i .

Non-Linear Filter. RAKAPOSHI uses as non-linear filtering function the 8-to-1 Boolean function obtained by considering the input as an element of the field $\mathbb{F}_2[x]/\langle p(x) \rangle \simeq \mathbb{F}_{2^8}$, where $p(x) = x^8 + x^4 + x^3 + x + 1$, and extracting the least significant bit of the result of the inverse operation in this field (with $0 \mapsto 0$). We note that this is the same function used as the non-affine component of the AES S-Box. This function $v(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ is a balanced Boolean function, with polynomial representation (ANF) of degree 7. We give the explicit polynomial expression of the function $v(\cdot)$ in the Appendix A.

In the RAKAPOSHI stream cipher, the input bits for the function v are extracted from both registers \mathcal{A} and \mathcal{B} , as

$$s_t = v(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128}).$$

We note that both the sets $B = \{b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128}\}$ and $A = \{a_{t+67}, a_{t+127}\}$ are full-positive difference sets [14].

4 Cipher Operation

In this section we describe the details of the cipher operation.

4.1 Keystream Generation

The cipher outputs one keystream bit at each cycle. Given the cipher state $\mathcal{S}^t = (\mathcal{A}^t, \mathcal{B}^t)$ at time t , the cipher operates as follows:

1. The keystream bit z_t is computed as $z_t = b_t \oplus a_t \oplus s_t$ and is output.
2. $c_0, c_1 \in \mathcal{A}^t$ are used to compute the updating function λ_B^t for \mathcal{B} .
3. Registers \mathcal{A} and \mathcal{B} are updated to obtain $\mathcal{A}^{t+1} = \lambda_A(\mathcal{A}^t)$ and $\mathcal{B}^{t+1} = \lambda_B^t(\mathcal{B}^t)$, respectively.

In typical operational mode, a fixed key and initialisation vector must not be used to produce more than 2^{64} keystream bits. Thus the cipher must be re-initialized (potentially by only modifying the IV) after at most 2^{64} cycles.

4.2 Initialisation Process

Before start producing the keystream, RAKAPOSHI goes through an initialisation process, in which the secret key and IV are loaded into the registers and mixed.

The secret key $\{k_0, k_1, \dots, k_{127}\}$ and IV $\{iv_0, iv_1, \dots, iv_{191}\}$ are loaded into the NLFSR and DLFSR, respectively, as follows:

$$\begin{aligned} a_0, a_1, \dots, a_{127} &\leftarrow k_0, k_1, \dots, k_{127}, \\ b_0, b_1, \dots, b_{191} &\leftarrow iv_0, iv_1, \dots, iv_{191}. \end{aligned}$$

The cipher then clocks 448 times with the output of the filter function $v(\cdot)$ (that is, s_t rather than z_t) being fed back into the cipher state. This process is divided into two stages:

- Stage 1

In the first stage of the initialisation, the cipher runs for 320 cycles, with the output of the non-linear filter function $v(\cdot)$ being fed back into the register \mathcal{B} . Thus during this stage the LFSR \mathcal{B} uses the following recurrence relation:

$$\begin{aligned} b_{t+192} = & b_t \oplus b_{t+14} \oplus b_{t+37} \oplus b_{t+41} \oplus b_{t+49} \oplus b_{t+51} \oplus b_{t+93} \oplus \\ & \overline{c_0} \cdot \overline{c_1} \cdot b_{t+107} \oplus \overline{c_0} \cdot c_1 \cdot b_{t+120} \oplus c_0 \cdot \overline{c_1} \cdot b_{t+134} \oplus \\ & c_0 \cdot c_1 \cdot b_{t+136} \oplus \overline{c_0} \cdot b_{t+155} \oplus c_0 \cdot b_{t+158} \oplus b_{176} \oplus s_t. \end{aligned}$$

– Stage 2

In the second stage of the initialisation, the cipher runs for further 128 cycles, with the output of the non-linear filter function $v(\cdot)$ being fed back into the register \mathcal{A} . Thus during this stage the NLFSR \mathcal{A} uses the following recurrence relation:

$$\begin{aligned} a_{t+128} = & 1 \oplus a_t \oplus a_{t+6} \oplus a_{t+7} \oplus a_{t+11} \oplus a_{t+16} \oplus a_{t+28} \oplus a_{t+36} \oplus a_{t+45} \oplus \\ & a_{t+55} \oplus a_{t+62} \oplus a_{t+7}a_{t+45} \oplus a_{t+11}a_{t+55} \oplus a_{t+7}a_{t+28} \oplus \\ & a_{t+28}a_{t+55} \oplus a_{t+6}a_{t+45}a_{t+62} \oplus a_{t+6}a_{t+11}a_{t+62} \oplus s_t. \end{aligned}$$

At the end of the initialisation, the cipher internal state is $\mathcal{S}^0 = (\mathcal{A}^0, \mathcal{B}^0)$, and it is ready to produce the first output bit of the keystream z_0 , as specified in Section [4.1](#).

5 Security Evaluation

In this section we provide a provisional security analysis of the RAKAPOSHI cipher, taking into account the most common cryptanalytic methods against stream ciphers.

5.1 Time-Memory Trade-Off Attacks

Time-Memory Trade-off attacks are generic attacks against several cryptographic constructions, consisting of a precomputation phase and an online phase. In a typical attack scenario, one would perform extensive offline computation and store the results, with the goal of reducing the time complexity of the online attack. Time-Memory Trade-off attacks were originally proposed by Hellman [\[18\]](#) for attacking the DES block cipher.

In the context of stream ciphers, TMTO attacks were first proposed by Babbage [\[2\]](#), and later improved by Biryukov and Shamir [\[7\]](#). In these attacks, one tries to invert the function mapping the cipher internal state to a segment of the keystream output. As a result, to prevent against such attacks a cipher should have its internal state at least twice as long as its key length.

On a different attack scenario, proposed by Hong and Sarkar [\[19,20\]](#), the function inverted maps the key/IV to a keystream segment. In their original approach, the IV is treated as part of the secret key, and as a result, prevention against such attack scenario requires stream ciphers to have the IV at least as long as the key. More recently Dunkelman and Keller [\[11\]](#) proposed a new approach, in which an attacker selects in advance several IVs, and mounts the attack to invert the function mapping the key to a keystream segment (with the chosen IVs). As a result, it follows that if n is the cipher key length, then the IV length must be at least $\frac{3}{2}n$ to withstand this form of TMTO attack.

The RAKAPOSHI stream cipher uses 128-bit keys and 192-bit IVs, with internal state with 320 bits. Therefore it withstands the currently known TMTO attacks against stream ciphers.

5.2 Guess-and-Determine Attacks

In guess-and-determine attacks, the attacker guesses a subset of the cipher internal state, and recovers further bits from the state based on the observed keystream and guessed values. This procedure may be repeated for other values until the full state and/or the secret key is recovered.

For the RAKAPOSHI stream cipher, one may guess a selected subset of the state of register \mathcal{A} and hope to recover bits from \mathcal{B} (since \mathcal{A} is the source of non-linearity to both registers). A naïve attack would be to guess all bits of the NLFSR \mathcal{A} , and then recover the state of the DLFSR \mathcal{B} . This attack is however essentially equivalent to a brute force attack on the cipher key, and is thus considered infeasible.

We have considered a number of alternative scenarios, in which an attacker guesses a number of bits of the cipher's internal state, and tries to recover further bits from the observed keystream. However due to the RAKAPOSHI internal structure (its registers' taps and input to non-linear filtering function), we have not been able to come up with an efficient attack. Thus we believe that RAKAPOSHI is not susceptible to guess-and-determine attacks.

5.3 Distinguishing Attacks

In distinguishing attacks, one attempts to find ways to distinguish the stream cipher (considered as a bit generator) from a purely random source of binary digits. It is notoriously difficult (if not impossible) to provide assurance that a cipher is not susceptible to distinguishing attacks. In fact some *secure* constructions (for instance, AES in counter mode) can be trivially distinguished from a random source [16], and as a result several researchers dispute the general applicability of some forms of distinguishing attacks against stream ciphers.

For RAKAPOSHI, we tried to construct linear distinguishers, by considering linear approximations of functions used in the cipher. For instance, the best affine approximation of the nonlinear filter has bias $\epsilon = 2^{-6}$, and the function uses as input 6 bits from register \mathcal{B} . For fixed values of c_0, c_1 , the number of terms in the register's recurrence function f is 11. Now, if we omit the 2-bit input from the NLFSR for the nonlinear filter, we can construct a linear distinguisher using 11 keystream bits as follows:

$$D : z_t \oplus z_{t+14} \oplus z_{t+37} \oplus z_{t+41} \oplus z_{t+49} \oplus z_{t+51} \oplus z_{t+93} \oplus z_{t+107} \oplus z_{t+155} \oplus z_{t+176} \oplus z_{t+192} = 0,$$

where we assume two clock bits c_0, c_1 from the NLFSR are $(c_0, c_1) = (0, 0)$ for each feedback function accompanying the distinguisher D . Using the piling-up lemma [26], the bias of the linear distinguisher is estimated as $2^{10} * (2^{-6})^{11} = 2^{-56}$. We note however that the two clock control bits in the linear recurrence f have to be determined to obtain the correct equation. As a result, the data complexity of this particular distinguishing attack for the cipher increases to $((1/2^{-56}) * 2^{2*6})^2 = 2^{136}$. Thus even by ignoring the effects of the NLFSR \mathcal{A} , this distinguishing attack can be considered infeasible.

We have also considered several other linear approximations, which are better than the above scenario; however, due to effects of the NLFSR, we have not

been able to find a more efficient linear approximation. Thus, we expect that the cipher is secure against distinguishing attacks.

5.4 Algebraic Attacks

Algebraic attacks against stream ciphers were originally proposed in 2003 by Courtois and Meier [9]. The attack is a powerful cryptanalytic technique against some LFSR-based stream cipher constructions (e.g. the filter generator). When mounting an algebraic attack, one attempts to construct a system of equations derived from the cipher operation, which can then be solved using a choice of methods. A stream cipher designer can protect the algorithm against algebraic attacks by using to compute the keystream, a non-linear filter function $z(\cdot)$ of high degree, for which the annihilating sets of both z or its complement $z + 1$ contain no low degree polynomials (the lowest degree of polynomials in either of these sets is called the *algebraic immunity* of z).

For the RAKAPOSHI stream cipher, the keystream output z_t is computed by means of a Boolean function over $(\mathbb{F}_2)^{10}$ given by

$$z_t = s_t + a_t + b_t = v(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128}) + a_t + b_t.$$

This function $z(\cdot)$ has degree 7 and algebraic immunity 4 (which is the same of the function v). Indeed, there are 21 linearly independent functions of degree 4 in the annihilating set of both z and $z + 1$. However both registers \mathcal{A} and \mathcal{B} are updated non-linearly (by functions g and f respectively). This increases very rapidly the degree and the complexity of the polynomial expression of the keystream output bits; this was verified experimentally for several clocks of the cipher. Thus conventional algebraic attacks do not seem to work against RAKAPOSHI.

We note an alternative to the approach above. One could guess several bits of the register \mathcal{A} , in the hope of keeping the resulting degrees at a reasonably low value (since \mathcal{A} is the source of non-linearity to both registers). A trivial, somewhat naïve algebraic attack against RAKAPOSHI would thus be to guess all the 128 values for the register \mathcal{A} , and generate 21 equations of degree 4 for each output bit of the cipher. If the state of register \mathcal{A} is fully known, then the complexity of an algebraic attack would be on the order of

$$\left(\sum_{k=0}^4 \binom{192}{k} \right)^3 \approx 2^{77}$$

operations, requiring approximately 2^{21} output bits. The total complexity of the attack would thus be on the order of 2^{205} operations. One could instead guess fewer bits from \mathcal{A} ; however the results above indicate that the overall complexity of the attack would remain well above the one required for exhaustive key search. Thus we conclude that RAKAPOSHI is secure against algebraic attacks.

5.5 Analysis of the Initialisation Process

The RAKAPOSHI initialisation process consists of 448 steps. The first 320 steps affect the subregister \mathcal{B} , ensuring that after this initial stage, all bits of the

Table 1. Diffusion of the Initialisation Process

	Stage 1 cycles			Stage 2 cycles	
	0	293	319	105	128
All registers of NLFSR	-	-	-	All KEY	All KEY
	-	-	-	-	All IV
All registers of DLFSR	-	All KEY	All KEY	All KEY	All KEY
	-	-	All IV	All IV	All IV

DLFSR \mathcal{B} have been influenced by all bits of the secret key (that is, the initial state of NLFSR \mathcal{A}) and all bits of the IV (that is, the initial state of DLFSR \mathcal{B}). The second stage consists of further 128 steps, which affects the subregister \mathcal{A} , and as a result all bits of the NLFSR \mathcal{A} would have been influenced by all bits of the IV and the secret key at the end of this stage. This is summarised in Table 1. We have also analysed how complex the relationship between the state bits and the secret key / IV are, by computing the algebraic expression of the state bits during several steps of the initialisation. Our experiments indicate that these are dense, high-degree polynomials involving a large number of internal state bits. Thus, the number of cycles in the initialisation process seems to be sufficient for diffusing the bits of the initial state in a very complex way, and we believe that RAKAPOSHI is secure against the most recent attacks on the initialisation process [10,25,32,35].

5.6 Statistical Tests

The statistical properties of the cipher depend on the properties of the output sequences of the NLFSR and DLFSR; given their construction, we expect the keystream of the cipher to have good statistical properties. We have evaluated the statistical properties for the cipher keystream, as well as the output sequences of the NLFSR and DLFSR using the NIST Test Suite [30]. The results indicated that the statistical properties of the RAKAPOSHI output sequence are good.

6 Implementation Aspects

In this section we discuss various aspects related to the implementation and performance of the RAKAPOSHI stream cipher.

6.1 Parallelization

The throughput of RAKAPOSHI can be increased by applying parallelization techniques. It is in fact possible to increase the cipher output to 64 bits per clock cycle by adding some additional circuits. To realise n times parallelization ($1 \leq n \leq 64$), we add circuits for n feedback functions (for $f(\cdot)$ and $g(\cdot)$), n keystream generation functions (for $v(\cdot)$), and add $n - 1$ bits to the NLFSR \mathcal{A} . The increase of registers for the DLFSR is not required for parallelization up to

Table 2. Evaluation Results of RAKAPOSHI

	Cir. Size (Slices)	Max. Cl. Freq. (MHz)	Throughput (Mbps)	Throughput / Slices
Spartan-II	199	111.6	111.6	0.56
Spartan-3	194	120.1	120.1	0.62
Spartan-3 (Opt.)	63	155.9	155.9	2.47
Vertex-II	193	218.9	218.9	1.13
Spartan-3 ($\times 8$)	342	125.5	1004.0	2.94
Spartan-3 ($\times 16$)	445	124.5	1992.0	4.48
Spartan-3 ($\times 32$)	849	125.0	4000.0	4.71
Spartan-3 ($\times 64$)	1302	95.0	6080.0	4.67

64 times, since the input for the nonlinear filter does not include any of the last 63 bits of DLFSR.

The design of RAKAPOSHI has been optimised for 32 times parallelization: the last 15 bits of the DLFSR are not used in the feedback function $f(\cdot)$. We note however that 64 times parallelization is also possible by a design criterion for NLFSR: the last 65 bits of NLFSR are not used for execution of the feedback function $g(\cdot)$.

6.2 Circuit Size and Performance

We present here the evaluation results of hardware implementations using an FPGA simulator. We implemented RAKAPOSHI targeted towards the Xilinx Spartan-II, Spartan-3, and Virtex-II FPGAs. We used Xilinx ISE 9.1 for post-place and route simulation and static timing analysis. Circuit sizes of the DLFSR, NLFSR, Nonlinear Filter function on Spartan-II are 1575 gates, 1060 gates, and 147 gates respectively. The circuit size of the whole algorithm is 3130 gates².

We have also implemented parallelized versions of the algorithm that produce 8-bit, 16-bit, 32-bit, and 64-bit outputs for each clock cycle. Table 2 shows evaluation results of circuit size, maximum clock frequency, throughput, and efficiency for each implementation. The efficiency is computed as throughput per slice. The efficiency improves due to circuit parallelization and it reaches its maximum by 32-times parallelization. The maximum throughput is approximately 6 Gbps using the 64-times parallelized circuit on Spartan-3.

6.3 Comparison with Other Stream Ciphers

As stated in Section 3, one of the design goals of the RAKAPOSHI stream cipher is to complement the eSTREAM portfolio in profile 2. As such, we present a comparison between various aspects of the RAKAPOSHI implementation and some of the eSTREAM ciphers in Table 3. We use in this comparison the evaluation results of circuit sizes and maximum clock frequency presented in [13][22].

² This corresponds to approximately one tenth of the circuit size of the K2 stream cipher [23], a predecessor of RAKAPOSHI which targets however software applications with high throughput requirements.

Table 3. Comparison on Spartan-3

Algorithm	Key Len	IV Len	Internal State	Cir. Size (Slices)	Max. Cl. Freq(MHz)	Thr/ Slices	# Cyc. for Init.
Grain [13]	80	64	160	122	193	1.58	160
Grain (opt.) [22]	80	64	160	44	196	4.45	160
Trivium [13]	80	80	288	188	206	1.10	1152
Trivium (opt.) [22]	80	80	288	50	240	4.80	1152
MICKEY-128 [13]	128	128	320	261	156	0.60	416
MICKEY-128(opt.) [22]	128	128	320	176	223	1.27	416
Grain-128(opt.) [22]	128	96	256	50	196	3.92	256
DECIM-128(opt.) [22]	128	128	352	89	43.5	0.49	1584
Rakaposhi	128	192	320	194	120.1	0.62	448
Rakaposhi (opt.)	128	192	320	63	155.9	2.47	448

We can note that RAKAPOSHI presents a competitive performance profile when compared with MICKEY-128 [5] with regards to throughput per slice. Grain v1 and Trivium are however more efficient than RAKAPOSHI. We note however that these algorithms do not provide 128-bit security level, which as noted earlier, we consider a particularly attractive feature when aiming for long-term deployments. Furthermore, given those ciphers’ state/key/IV sizes, they would appear to be susceptible to more recently proposed TMTO attacks (see Section 5.1). We note that the latter remark is also applicable to Grain-128 [17].

7 Conclusion

In this paper we propose a new stream cipher, called RAKAPOSHI. The algorithm presents a simple and elegant design, and is particularly suitable for hardware applications with restricted resources. We have presented a provisional security evaluation of the cipher, which indicates that it is secure against the most common attacks against stream ciphers. Furthermore, evaluation of several implementation and performance aspects shows that RAKAPOSHI is a competitive alternative to ciphers in the final eSTREAM portfolio. We believe therefore that RAKAPOSHI can complement the current eSTREAM portfolio in profile 2, while presenting some attractive additional features, increasing thus the choice of secure lightweight stream ciphers suitable for hardware applications with restricted resources.

References

1. Arnault, F., Berger, T., Lauradoux, C.: F-FCSR Stream Ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) *New Stream Cipher Designs*. LNCS, vol. 4986, pp. 170–178. Springer, Heidelberg (2008)
2. Babbage, S.: Improved “exhaustive search” attacks on stream ciphers. In: *IEE European Convention on Security and Detection*, vol. 408, pp. 161–165 (1995)

3. Babbage, S., De Canniere, C., Canteaut, A., Cid, C., Gilbert, H., Johansson, T., Parker, M., Preneel, B., Rijmen, V., Robshaw, M.: The eSTREAM Portfolio (rev.1), September 08 (2008), http://www.ecrypt.eu.org/stream/portfolio_revision1.pdf
4. Babbage, S., De Canniere, C., Canteaut, A., Cid, C., Gilbert, H., Johansson, T., Parker, M., Preneel, B., Rijmen, V., Robshaw, M.: The eSTREAM Portfolio, April 15 (2008), <http://www.ecrypt.eu.org/stream/portfolio.pdf>
5. Babbage, S., Dodd, M.: The MICKEY Stream Ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) *New Stream Cipher Designs*. LNCS, vol. 4986, pp. 191–209. Springer, Heidelberg (2008)
6. Beth, T., Piper, F.C.: The stop-and-go-generator. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) *EUROCRYPT 1984*. LNCS, vol. 209, pp. 88–92. Springer, Heidelberg (1985)
7. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In: Okamoto, T. (ed.) *ASIACRYPT 2000*. LNCS, vol. 1976, pp. 1–13. Springer, Heidelberg (2000)
8. Cid, C., Robshaw, M.: The eSTREAM Portfolio 2009 Annual Update, July 31 (2009), http://www.ecrypt.eu.org/stream/eStream_reportJul09.pdf
9. Courtois, N., Meier, W.: Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (ed.) *EUROCRYPT 2003*. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
10. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) *FroCoS 2009*. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
11. Dunkelmann, O., Keller, N.: Treatment of the Initial Value in Time-Memory-Data Tradeoff Attacks on Stream Ciphers. *Information Processing Letters* 107, 133–137 (2008)
12. eSTREAM, the ECRYPT Stream Cipher Project, <http://www.ecrypt.eu.org/stream/>
13. Gaj, K., Southern, G., Bachimanchi, R.: Comparison of hardware performance of selected Phase II eSTREAM candidates. In: *Proceedings of SASC 2007*, Bochum (2007)
14. Golic, J.D.: On Security of Nonlinear Filter Generators. In: Gollmann, D. (ed.) *FSE 1996*. LNCS, vol. 1039, pp. 173–188. Springer, Heidelberg (1996)
15. Hell, M., Johansson, T.: Breaking the F-FCSR-H Stream Cipher in Real Time. In: Pieprzyk, J. (ed.) *ASIACRYPT 2008*. LNCS, vol. 5350, pp. 557–569. Springer, Heidelberg (2008)
16. Hell, M., Johansson, T., Brynielsson, L.: An overview of distinguishing attacks on stream ciphers. *Cryptography and Communications* 1(1), 71–94 (2009)
17. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: *Proceedings of 2006 IEEE International Symposium on Information Theory*, pp. 1614–1618. IEEE, Los Alamitos (2006)
18. Hellman, M.E.: A Cryptanalytic Time-Memory Tradeoff. *IEEE Transactions on Information Theory* 26(4), 401–406 (1980)
19. Hong, J., Sarkar, P.: New Applications of Time Memory Data Tradeoffs. In: Roy, B. (ed.) *ASIACRYPT 2005*. LNCS, vol. 3788, pp. 353–372. Springer, Heidelberg (2005)
20. Hong, J., Sarkar, P.: Rediscovery of Time Memory Tradeoffs. *Cryptology ePrint Archive*, Report 2005/090 (2005), <http://eprint.iacr.org/>

21. Horan, D., Guinee, R.: A Novel Keystream Generator using Pseudo Random Binary Sequences for Cryptographic Applications. In: Irish Signals and Systems Conference 2006, pp. 451–456. IEEE, Los Alamitos (2006)
22. Hwang, D., Chaney, M., Karanam, S., Ton, N., Gaj, K.: Comparison of FPGA-Targeted Hardware Implementations of eSTREAM Stream Cipher Candidates. In: Proceedings of SASC (2008); Lausanne
23. Kiyomoto, S., Tanaka, T., Sakurai, K.: FPGA-Targeted Hardware Implementations of K2. In: Proceedings of SECURE 2008, pp. 270–277 (2008)
24. Kiyomoto, S., Tanaka, T., Sakurai, K.: K2: A Stream Cipher Algorithm Using Dynamic Feedback Control. In: Proceedings of SECURE 2007, pp. 204–213 (2008)
25. Lee, Y., Jeong, K., Sung, J., Hong, S.: Related-Key Chosen IV Attacks on Grain-v1 and Grain-128. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 321–335. Springer, Heidelberg (2008)
26. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
27. Medina, M., Domínguez, A.: Caracterización de Secuencias Binarias Pseudoaleatorias generadas mediante LFSR con Realimentación Dinámica (DLFSR). In: Proceedings of XVIII Simposium Nacional de la URSI, A Coruña, Spain (2003)
28. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
29. Mita, R., Palumbo, G., Pennisi, S., Poli, M.: Pseudorandom bit generator based on dynamic linear feedback topology. *Electronic Letters* 28(19), 1097–1098 (2002)
30. National Institute of Standards and Technology. NIST Statistical Test, <http://csrc.nist.gov/rng/>
31. Robshaw, M., Billet, O.: New Stream Cipher Designs. LNCS, vol. 4986. Springer, Heidelberg (2008)
32. Khazaei, S., Fischer, S., Meier, W.: Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)
33. Simpson, L.R., Dawson, E., Golic, J., Millan, W.: LILI Keystream Generator. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 248–261. Springer, Heidelberg (2001)
34. Soriano, M.: Stream ciphers based on NLFSR. In: Proceedings of SBT/IEEE International Telecommunications Symposium 1998, pp. 528–533. IEEE, Los Alamitos (1998)
35. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack, *Cryptology ePrint archive*, report 2007/413 (2007)

Appendix

A Rakaposhi Non-linear Function

The RAKAPOSHI stream cipher uses the following Boolean function to produce the keystream output.

$$\begin{aligned}
 v(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = & \\
 & x_0x_1x_2x_3x_4x_5x_6 + x_0x_1x_2x_3x_4x_5 + x_0x_1x_2x_3x_4x_6 + x_0x_1x_2x_3x_5x_6x_7 + \\
 & x_0x_1x_2x_3x_5x_6 + x_0x_1x_2x_3x_5x_7 + x_0x_1x_2x_3x_5 + x_0x_1x_2x_3x_6x_7 + \\
 & x_0x_1x_2x_4x_5x_6 + x_0x_1x_2x_4 + x_0x_1x_2x_5x_6 + x_0x_1x_2x_5x_7 + x_0x_1x_2x_7 + \\
 & x_0x_1x_2 + x_0x_1x_3x_4x_5x_6x_7 + x_0x_1x_3x_4x_5x_7 + x_0x_1x_3x_4x_5 + x_0x_1x_3x_4x_7 + \\
 & x_0x_1x_3x_4 + x_0x_1x_3x_6 + x_0x_1x_4x_5x_6x_7 + x_0x_1x_4x_5x_6 + x_0x_1x_4x_5x_7 + \\
 & x_0x_1x_4x_6x_7 + x_0x_1x_4x_7 + x_0x_1x_5x_6x_7 + x_0x_1x_5x_6 + x_0x_1x_5 + x_0x_1x_6 + \\
 & x_0x_1 + x_0x_2x_3x_4x_5x_6 + x_0x_2x_3x_4x_5x_7 + x_0x_2x_3x_4 + x_0x_2x_3x_5x_6x_7 + \\
 & x_0x_2x_3x_5x_6 + x_0x_2x_3x_5x_7 + x_0x_2x_3x_6 + x_0x_2x_4x_5x_6x_7 + x_0x_2x_5x_6 + \\
 & x_0x_2x_5 + x_0x_2x_6x_7 + x_0x_2x_7 + x_0x_3x_4x_5x_6x_7 + x_0x_3x_4x_5x_6 + x_0x_3x_4x_5x_7 + \\
 & x_0x_3x_4x_5 + x_0x_3x_4x_7 + x_0x_3x_5x_6x_7 + x_0x_3x_5 + x_0x_3x_6 + x_0x_3 + x_0x_4x_5x_6 + \\
 & x_0x_4x_6x_7 + x_0x_5x_6 + x_0x_6 + x_0 + x_1x_2x_3x_4 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_5x_7 + \\
 & x_1x_2x_3x_5 + x_1x_2x_3 + x_1x_2x_4x_5x_6 + x_1x_2x_4x_6 + x_1x_2x_4 + x_1x_2x_5 + x_1x_2 + \\
 & x_1x_3x_4x_5x_6x_7 + x_1x_3x_4x_5x_7 + x_1x_3x_4x_6x_7 + x_1x_3x_4x_6 + x_1x_3x_4 + \\
 & x_1x_3x_5x_6 + x_1x_3x_5 + x_1x_3x_6 + x_1x_3x_7 + x_1x_4x_5x_6x_7 + x_1x_4x_5x_7 + \\
 & x_1x_5x_6 + x_1x_5x_7 + x_1x_5 + x_1x_6x_7 + x_1x_6 + x_1 + x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_7 + \\
 & x_2x_3x_4x_5 + x_2x_3x_4x_6x_7 + x_2x_3x_4 + x_2x_3x_5x_7 + x_2x_3x_6x_7 + x_2x_3x_6 + \\
 & x_2x_4x_5x_6 + x_2x_4x_5x_7 + x_2x_4x_5 + x_2x_4x_6x_7 + x_2x_4x_6 + x_2x_4x_7 + x_2x_4 + \\
 & x_2x_5x_6x_7 + x_2x_6x_7 + x_2x_6 + x_2x_7 + x_3x_4x_5x_6x_7 + x_3x_4x_5 + x_3x_4x_6x_7 + \\
 & x_3x_4x_6 + x_3x_4x_7 + x_3x_5x_6x_7 + x_3x_6x_7 + x_3x_6 + x_3x_7 + x_4x_5x_6 + x_4x_5 + \\
 & x_5x_6x_7 + x_5x_6 + x_5 + x_6 + x_7.
 \end{aligned}$$

B Test Vector

```

Key: 00000000000000000000000000000000
IV: 0000000000000000000000000000000000000000000000000000000000000000
Internal State Bits after the Initialization:
(NLFSR-A) 3c12b227eccb28a0baf327a7d42a51e5
(DLFSR-B) 619344585ae94087412e9863bd028f18f42eefe6378c5011
Keystream:
7a72bd702002121880960ed4ae0c054ecad09b0459c334866fbbd8
84aa0ff5585497943c6095d427c96eeb8719f87a02761465d0f62a
1e0faad849302104827e6db2e0b81e49a7b81ce170e4cf261468d6
6b2e6e13cfcabca1073f2077298b2c0fe0da1feb8c1e20b27f5907
b883eb17c5165113acfb2a7ca7a0c6cf3578f87c

```

Design of Reliable and Secure Multipliers by Multilinear Arithmetic Codes

Zhen Wang¹, Mark Karpovsky¹, Berk Sunar^{2,*}, and Ajay Joshi¹

¹ Boston University, Reliable Computing Laboratory, 8 Saint Marys Street, Boston, MA, USA
{lark,markkar,joshi}@bu.edu

² Worcester Polytechnic Institute, CRIS Laboratory, 100 Institute Road, Worcester, MA 01609
sunar@wpi.edu

Abstract. We propose an efficient technique for the detection of errors in cryptographic circuits introduced by strong adversaries. Previously a number of linear and nonlinear error detection schemes were proposed. Linear codes provide protection only against primitive adversaries which no longer represents practice. On the other hand nonlinear codes provide protection against strong adversaries, but at the price of high area overhead (200–300%). Here we propose a novel error detection technique, based on the random selection of linear arithmetic codes. Under mild assumptions the proposed construction achieves near nonlinear code error detection performance at a lower cost (about 50% area overhead) due to the fact that no nonlinear operations are needed for the encoder and decoder.

1 Introduction

Cryptographic devices are vulnerable to side-channel attacks such as timing analysis attacks [1], power analysis attacks [2] and fault injection attacks [3],[4]. Due to their active and adaptive nature, fault based attacks are one of the most powerful types of side-channel attacks. Since a fault attack was demonstrated by Boneh et al. in [5] in 1996, numerous papers have been published proposing a variety of fault attacks on both public-key and private-key cryptographic devices. One of the most efficient fault injection attacks on AES-128, for example, requires only two faulty ciphertexts to retrieve all 128 bits of the secret key [6]. Without proper protection architecture against fault injection attacks, the security of cryptographic devices can never be guaranteed.

Error detecting codes are often used in cryptographic devices to detect errors caused by injected faults and prevent the leakage of useful information to attackers. Most of the proposed error detecting codes are linear codes like parity codes, Hamming codes and AN codes [7]. Protection architectures based on linear codes concentrate their error detecting abilities on errors with small multiplicities or errors of particular types, e.g. errors with odd multiplicities or byte errors. However, in the presence of unanticipated types of errors linear codes can provide little protection. Linear parity codes, for example, can detect no errors with even multiplicities. By carefully selecting faults and

* This author's work is based upon work supported by the National Science Foundation under Grant No. CNS-0831416.

injection methods an attacker can with high probability bypass the protection architectures based on linear codes and still be able to break the security of cryptographic devices in a reasonably short time.

In [8], robust algebraic codes were proposed as an alternative to classical linear codes to protect cryptographic devices implementing AES against fault injection attacks. In [9], robust arithmetic residue codes were proposed which can be used to design fault tolerant cryptographic devices performing arithmetic operations. Instead of concentrating the error detecting abilities on particular types of errors, robust codes provide nearly equal protection against all error patterns. Hence robust codes eliminate the weakness of linear codes which can be exploited by attackers to mount successful fault attacks. Variants of both algebraic and arithmetic robust codes – partially robust and minimum distance robust codes – were proposed in [10]. These architectures allow various trade-offs in terms of robustness and hardware overhead.

Robust codes are based on nonlinear functions [11] and the robustness of the code is highly related to the nonlinearity of the function. Systematic robust codes, for example, can be constructed by appending a signature generated by a nonlinear function f to the information part of the code. The worst case error masking probability of any nonzero error is bounded by P_f , which gives a measure of the degree of the nonlinearity of f .

The main disadvantage of robust codes is the large hardware overhead when implementing nonlinear operations for the encoding and decoding circuits. In this paper, we propose a different method to achieve similar levels of protection. Instead of using nonlinear functions to generate the signature of the code, we randomly select a code from multiple linear codes at each clock cycle. The resulting codes are called **multilinear codes**. The proposed method can have as small number of undetectable errors as classical robust codes while requiring much less hardware overhead due to the fact that no nonlinear operations are needed for the encoder and decoder.

Multipliers are widely used as sub-blocks in public key cryptosystems. In this paper we present constructions of multilinear arithmetic codes and their use to design reliable multipliers. We compare the proposed protection architecture for multipliers with that based on single linear arithmetic code. We assume that countermeasures are implemented in the cryptographic device preventing the attackers from tampering with the clock signals. We further assume that a low-rate true random number generator (e.g. [12]) is available. In fact, most cryptographic devices incorporate a true random number generator by default for key initialization, random pad computation, challenge generation etc. The error detection capabilities of different architectures are simulated in MATLAB and the advantage of the proposed technique is demonstrated.

The constructions of multilinear algebraic codes and the analysis of fault detection capabilities of architectures based on multilinear algebraic codes were discussed in [13].

The paper is organized as follows. Section 2 describes the error and attacker models we use throughout the paper. In section 3 we formalize the design and propose several constructions based on randomly selecting multiple linear codes. In Section 4 we compare different protection architectures for a fixed precision multiplier. We finish the paper by drawing the conclusions in Section 5.

2 Error and Attacker Model

In this paper we concentrate on the analysis of the error detection abilities for systematic arithmetic codes and the reliability of multipliers based on these codes. Different from the widely used nonsystematic *AN* codes [7], the codewords of systematic arithmetic codes contain two parts: the information part and the redundancy part. Any codeword c can be written in the format of (x, y) , $x \in Z_{2^k}$, $y \in Z_{2^r}$, where k is the number of information bits, r is the number of redundant bits and Z_{2^k} is the additive group of integers $\{0, 1, \dots, 2^k - 1\}$. We denote by $e = (e_x, e_y)$ the error vector and $\tilde{c} = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$ the distorted codeword in which $e_x \in Z_{2^k}$, $e_y \in Z_{2^r}$, $+$ is the arithmetic addition and $| \cdot |_p$ is the modulo p operation.

Let C be an arithmetic code. An error $e = (e_x, e_y)$ is masked by a codeword $c = (x, y) \in C$ if $\tilde{c} = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$ also belongs to C . Given an error e , the error masking probability $Q(e)$ is calculated as follows:

$$Q(e) = \frac{|\{c | c \in C, \tilde{c} \in C\}|}{|C|}. \quad (1)$$

If an error is masked by all codewords of the code, $Q(e) = 1$ and the error is called **undetectable**. If $0 < Q(e) < 1$, the error is called **conditionally detectable**. Different from algebraic codes, arithmetic codes usually do not have undetectable errors. To illustrate the advantage of the proposed codes, we compare the number of **bad errors**, which are errors e with $Q(e) \geq 0.5$, for linear arithmetic codes and the proposed multilinear arithmetic codes. Since bad errors are the most difficult to detect, we will show that the transition from linear to multilinear arithmetic codes results in a drastic reduction of the number of bad errors and an improvement of the error detection ability of the code.

Throughout the paper we assume a strong attacker model in which an attacker knows everything about the hardware architecture of the device including the code used to detect errors. The attacker can utilize any fault injection methodologies and is able to inject faults with high spatial resolution to generate a specific error vector (e_x, e_y) at the output of the devices. The only limitation on the attacker is that he cannot change the error at each clock cycle. Once faults are injected and an error is generated, the faults stay for several clock cycles and the error tends to repeat. This is the case for several well known fault injection methodologies such as introducing power glitches into the power supply, using laser guns, etc [4]. We call this kind of channels where errors have high probabilities to repeat themselves for several consecutive clock cycles **lazy channels** or **channels with memory**.

The advantages of multilinear arithmetic codes in terms of error detection capabilities are two-fold. First, they are better than linear arithmetic codes in a sense that it has much smaller number of bad errors. Second, multilinear arithmetic codes have much higher error detection abilities than linear codes in lazy channels hence they will effectively prevent the attacker from implementing a successful fault induction attack under the aforementioned attacker model. To facilitate the analysis and comparison of the error protection architectures based on different codes in lazy channels, we assume that errors last for at least t consecutive clock cycles, $t \geq 1$.

The experimental results for the error detection properties of multipliers protected by different codes are presented in Section 4 which shows that as t increases the error detection probabilities are much higher for architectures based on multilinear arithmetic codes than that based on single linear arithmetic code.

3 Constructions

We first analyze the error detection properties of linear arithmetic codes.

Theorem 1. (Linear Arithmetic Codes) *Let C be a linear arithmetic code defined by*

$$C = \{(x, y) | x \in Z_{2^k}, y = f(x) \in Z_{2^r}\}, \quad (2)$$

where p is a prime number larger than 2 and $f(x) = |x|_p$ where $| \cdot |_p$ represents the modulo p reduction operation. Denote by $e = (e_x, e_y)$ an additive error; $e_x \in Z_{2^k}, e_y \in Z_{2^r}, r = \lceil \log_2 p \rceil, c + e = (|x + e_x|_{2^k}, |y + e_y|_{2^r}), c \in C$. When $p \ll 2^k$, the number of bad errors is upper bounded by

$$2 \cdot (2^k + p - 2^k (H_{p-1} - H_{\lfloor \frac{p}{2} \rfloor}) - \frac{2^{k-1}}{p}). \quad (3)$$

In the equation H_n represents the n -th harmonic number. For large p the difference $H_{p-1} - H_{\lfloor \frac{p}{2} \rfloor}$ converges to $\ln 2$. Thus, for large p , the number of bad errors is upper bounded by $2p - 2^{k+1}(\ln 2 - 1) - 2^k/p$.

If no errors occur to the redundant part of the code, the number of bad errors occurring only to the information part of the code is upper bounded by

$$2 \cdot \lceil \frac{2^{k-1}}{p} \rceil. \quad (4)$$

When p is large and $p \ll 2^k$, the estimated probability of bad errors is $0.3 \cdot 2^{-r+1}$. The estimated probability of bad errors occurring to the information part is 2^{-2r} .

Proof. To simplify the analysis, we divide the errors into two classes according to the value of $x + e_x$.

1. $x + e_x < 2^k$, we have $|x + e_x|_{2^k} = x + e_x, f(x + e_x) = |x + e_x|_p$.
 - (a) $|x|_p + e_y < p$, then $||x|_p + e_y|_{2^r} = |x|_p + e_y$. An error (e_x, e_y) is masked if and only if $|x + e_x|_p = |x|_p + e_y$. Or equivalently $|e_x|_p = e_y$. For a codeword x to mask a given error (e_x, e_y) , the following conditions must be satisfied:

$$x + e_x < 2^k, \quad (5)$$

$$|x|_p + e_y < p, \quad (6)$$

$$|e_x|_p = e_y. \quad (7)$$

From (6) and (7) we have $|x|_p < p - |e_x|_p$. For a certain value of $|x|_p < p - |e_x|_p$, the number of x satisfying (5) is bounded by $\lceil \frac{2^k - e_x}{p} \rceil$. Thereby for

a given error (e_x, e_y) , the total number of codewords that mask the error is $\lceil \frac{2^k - e_x}{p} \rceil \cdot (p - |e_x|_p)$. For bad errors the error masking probability is larger or equal to 0.5. Thus

$$2^{-k} \cdot \lceil \frac{2^k - e_x}{p} \rceil \cdot (p - |e_x|_p) \geq 0.5. \quad (8)$$

When $p \ll 2^k$, it is reasonable to rewrite (8) as follows:

$$2^{-k} \cdot \frac{2^k - e_x}{p} \cdot (p - |e_x|_p) \geq 0.5. \quad (9)$$

Thereby,

$$e_x \leq \frac{2^{k-1}(p - 2 \cdot |e_x|_p)}{p - |e_x|_p}. \quad (10)$$

We know that $e_x \geq 0$, so

$$0 \leq |e_x|_p \leq \lfloor \frac{p}{2} \rfloor. \quad (11)$$

The total number of e_x satisfying (10) and (11) is upper bounded by (For simplicity, let $i = |e_x|_p$.)

$$\sum_{i=0}^{\lfloor \frac{p}{2} \rfloor} \left(\frac{1}{p} \cdot \frac{2^{k-1}(p - 2i)}{p - i} + 1 \right). \quad (12)$$

So the number of bad errors in this class is bounded by (12), which can be simplified to be

$$\frac{2^k + p}{p} \left(\lfloor \frac{p}{2} \rfloor + 1 \right) - 2^{k-1} \cdot \sum_{i=0}^{\lfloor \frac{p}{2} \rfloor} \left(\frac{1}{p - i} \right). \quad (13)$$

- (b) $p \leq |x|_p + e_y < 2^r$, errors in this class will never be masked because the redundant part is a value that cannot occur.
- (c) $|x|_p + e_y \geq 2^r$, then $||x|_p + e_y|_{2^r} = |x|_p + e_y - 2^r$. An error (e_x, e_y) is masked if and only if $|x + e_x|_p = |x|_p + e_y - 2^r$. Or equivalently $|e_x|_p = |e_y - 2^r|_p$. It is easy to show that $e_y - 2^r \in [-p + 1, 0]$, so $|e_y - 2^r|_p = p + e_y - 2^r$. For a codeword x to mask a given error (e_x, e_y) , the following conditions must be satisfied:

$$x + e_x < 2^k, \quad (14)$$

$$|x|_p + e_y \geq 2^r, \quad (15)$$

$$|e_x|_p = e_y + p - 2^r. \quad (16)$$

From (15) and (16) we have $|x|_p \geq p - |e_x|_p$. For a certain value of $|x|_p \geq p - |e_x|_p$, the number of x satisfying (14) is bounded by $\lceil \frac{2^k - e_x}{p} \rceil$. Thereby for a given error (e_x, e_y) , the total number of codewords that mask the error is

$\lceil \frac{2^k - e_x}{p} \rceil \cdot |e_x|_p$. For bad errors the error masking probability is larger or equal to 0.5. Thus

$$2^{-k} \cdot \lceil \frac{2^k - e_x}{p} \rceil \cdot |e_x|_p \geq 0.5. \quad (17)$$

When $p \ll 2^k$, we rewrite (17) as follows:

$$2^{-k} \cdot \frac{2^k - e_x}{p} \cdot |e_x|_p \geq 0.5. \quad (18)$$

So

$$e_x \leq \frac{1}{|e_x|_p} \cdot 2^{k-1} \cdot (2|e_x|_p - p). \quad (19)$$

Because $e_x \geq 0$,

$$|e_x|_p \geq \lceil \frac{p}{2} \rceil. \quad (20)$$

The total number of e_x satisfying (19) and (20) is upper bounded by (For simplicity, let $i = |e_x|_p$.)

$$num < \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} (\frac{1}{p} \cdot \frac{2^{k-1}(2i-p)}{i} + 1). \quad (21)$$

So the number of bad errors in this class is bounded by (21), which can be simplified to be

$$\frac{2^k + p}{p} \cdot (p - \lceil \frac{p}{2} \rceil) - 2^{k-1} \cdot \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i}. \quad (22)$$

From (13) and (22), the total number of bad errors for the case when $x + e_x < 2^k$ is bounded by $2^k + p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}$.

2. $x + e_x \geq 2^k$, we have $|x + e_x|_{2^k} = x + e_x - 2^k$, $f(x + e_x) = |x + e_x - 2^k|_p$. Following the same analysis, we can show that the number of bad errors in this class is also bounded by $2^k + p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}$.

Thereby for linear arithmetic codes, an upperbound of the number of bad errors is

$$\begin{aligned} & 2 \cdot (2^k + p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}) \\ & = 2 \cdot (2^k + p - 2^k (H_{p-1} - H_{\lceil \frac{p}{2} \rceil}) - \frac{2^{k-1}}{p}). \end{aligned}$$

If no errors occur to the redundant part of the code, $e_y = 0$. For the case when $x + e_x < 2^k$, a codeword x mask an error $e = (e_x, e_y = 0)$ if and only if $|e_x|_p = e_y = 0$. It is easy to prove that the number of errors in this class is upper bounded by $\lceil \frac{2^{k-1}}{p} \rceil$. Similarly, when $x + e_x \geq 2^k$, the number of bad errors in the format of $(e_x, 0)$ is also upper

bounded by $\lceil \frac{2^{k-1}}{p} \rceil$. So the total number of bad errors occurring to the information part of the code is no more than $2 \cdot \lceil \frac{2^{k-1}}{p} \rceil$.

When p is large and $p \ll 2^k$, the estimated probability of bad errors and bad errors occurring to the information part of the code can be derived directly from (3) and (4). ■

The number of bad errors occurring to the information part of the code decreases as p increases according to (4). When $p > 2^{k-1}$, there are nearly no bad errors in the format of $(e_x, e_y = 0)$. However, the total number of bad errors is still very large for linear arithmetic codes.

In general, the hardware overhead for the encoder of the code is mostly affected by the number of redundant bits $r = \lceil \log_2(p) \rceil$. Many different p can be selected when r is fixed and some of them are better in terms of the total number of bad errors. Table 1 shows the best p and the corresponding fraction of bad errors for different k and r . The numbers outside the parentheses are the prime numbers which will result in the smallest number of bad errors for the given k and r . The numbers inside the parentheses are the corresponding fractions of bad errors.

When $2^r \ll 2^k$, we should select p to be the largest possible prime number for the purpose of minimizing the number of bad errors, e.g. when $r = 4$, the best p for the three k values are all 13, which is the largest prime number less than 2^4 . However, if r is comparable to k , smaller p can achieve the same error detection capability as larger p .

The smallest fraction of bad errors for linear arithmetic code is of the order of 2^{-r} . The only way to reduce the fraction is to increase the number of redundant bits, which is costly in terms of the hardware overhead.

We next propose a construction of multilinear arithmetic codes that have smaller total number of bad errors than linear codes with the same number of redundant bits r .

Table 1. Best p and the corresponding fractions of bad errors (in parentheses) for different k and r for linear arithmetic codes

	$k = 16$	$k = 32$	$k = 64$
$r = 4$	13 ($2^{-4.6}$)	13 ($2^{-4.6}$)	13 ($2^{-4.6}$)
$r = 8$	131 ($2^{-8.6}$)	251 ($2^{-8.6}$)	251 ($2^{-8.6}$)
$r = 12$	2053 ($2^{-12.5}$)	2053 ($2^{-12.5}$)	4093 ($2^{-12.5}$)
$r = 16$	—	32771 ($2^{-16.7}$)	65521 ($2^{-16.7}$)
$r = 20$	—	524309 ($2^{-20.7}$)	1048549 ($2^{-20.7}$)

Theorem 2. ($(\llbracket x \rrbracket_p, \llbracket 2x \rrbracket_p)$ **Multilinear Code**) Let C_1, C_2 be two arithmetic systematic codes defined by

$$C_i = \{(x, y) | x \in Z_{2^k}, y = f_i(x) \in Z_{2^r}\}, i \in \{1, 2\},$$

where $f_1(x) = \llbracket x \rrbracket_p$, $f_2(x) = \llbracket 2x \rrbracket_p$, p is a prime number larger than 2 and $\llbracket \cdot \rrbracket_p$ is the modulo operation. Denote by $e = (e_x, e_y)$ the arithmetic errors and $\tilde{c} = c + e = (\llbracket x + e_x \rrbracket_{2^k}, \llbracket y + e_y \rrbracket_{2^r})$ the distorted codeword, where $e_x \in Z_{2^k}, e_y \in Z_{2^r}, r = \lceil \log_2 p \rceil$

is the number of redundant bits. If we randomly select C_1 and C_2 to encode the original messages with equal probability, the total number of bad errors is upper bounded by

$$2 \cdot \left\lceil \frac{2^{k-1}}{p} \right\rceil. \quad (23)$$

When $p \ll 2^k$, the estimated probability of bad errors for $[|x|_p, |2x|_p]$ multilinear codes is 2^{-2r} .

Proof. For each linear code, there are four cases resulting in four sets of bad errors (refer to the proof for Theorem 11), which are shown in Table 2. When we randomly select C_1 and C_2 with equal probability, an error $e = (e_x, e_y)$ is bad if and only if the total number of codewords in C_1 and C_2 that mask e is larger or equal to 2^k .

Table 2. Classification of bad errors for linear arithmetic codes

Case1	Case2	Case3	Case4
$x + e_x < 2^k$	$x + e_x < 2^k$	$x + e_x \geq 2^k$	$x + e_x \geq 2^k$
$f_i(x) + e_y < p$	$f_i(x) + e_y \geq 2^r$	$f_i(x) + e_y < p$	$f_i(x) + e_y \geq 2^r$
$f_i(e_x) = e_y$	$f_i(e_x) = e_y + p - 2^r$	$f_i(e_x - 2^k) = e_y$	$f_i(e_x - 2^k) = e_y + p - 2^r$

1. For a given error e , when C_1 is in Case1 (i.e. $x + e_x < 2^k, f_1(x) + e_y < p, f_1(e_x) = e_y$) or Case2 and C_2 is in Case3 or Case4, the total number of codewords masking the error e is less than 2^k . So there are no bad errors in this class. Similarly, when C_1 is in Case3 or Case4 and C_2 is in Case1 or Case2, there are no bad errors.
2. C_1 is in Case1 and C_2 is in Case2. For $C_1, |x|_p + e_y < p$, the possible number of $|x|_p$ is $p - e_y$. For $C_2, |2x|_p + e_y \geq 2^r$, the possible number of $|x|_p$ is $p - 2^r + e_y$. For each possible value of $|x|_p$, the number of x is $\lceil \frac{2^k - e_x}{p} \rceil$. It is easy to prove that the total number of x masking the error is less than 2^k . So there are no bad errors in this class. Similarly we can prove that for the following three cases there are also no bad errors.
 - (a) C_1 is in Case2, C_2 is in Case1;
 - (b) C_1 is in Case3, C_2 is in Case4;
 - (c) C_1 is in Case4, C_2 is in Case3.
3. When C_1 and C_2 both belong to Case2, $x + e_x < 2^k$, we have $|x + e_x|_{2^k} = x + e_x, |x|_p + e_y \geq 2^r$ and $|2x|_p + e_y \geq 2^r$. In this case $||x|_p + e_y|_{2^r} = |x|_p + e_y - 2^r, ||2x|_p + e_y|_{2^r} = |2x|_p + e_y - 2^r$. For C_1 , an error (e_x, e_y) is missed if and only if

$$|x + e_x|_p = |x|_p + e_y - 2^r. \quad (24)$$

Equivalently,

$$|e_x|_p = |e_y - 2^r|_p. \quad (25)$$

For C_2 , an error (e_x, e_y) is missed if and only if

$$|2 \cdot (x + e_x)|_p = |2x|_p + e_y - 2^r. \quad (26)$$

Thus

$$|2e_x|_p = |e_y - 2^r|_p. \quad (27)$$

From (25) and (27) we have $|e_x|_p = |e_y - 2^r|_p = e_y + p - 2^r = 0$. For an error to be masked by both of the codes, the following conditions must be satisfied:

$$x + e_x < 2^k, \quad (28)$$

$$|x|_p + e_y \geq 2^r, \quad (29)$$

$$|2x|_p + e_y \geq 2^r, \quad (30)$$

$$|e_x|_p = e_y + p - 2^r = 0. \quad (31)$$

From (31), $e_y = 2^r - p$. So $|x|_p + e_y < 2^r$, $|2x|_p + e_y < 2^r$. Thereby no errors in this case will be masked by both of the codes. Errors in this class are all non-bad errors. Similarly, when C_1 and C_2 both belong to Case4, there are no bad errors.

4. When C_1 and C_2 both belong to Case1, $x + e_x < 2^k$, we have $|x + e_x|_{2^k} = x + e_x$, $f_1(x + e_x) = |x + e_x|_p$, $f_2(x + e_x) = |2 \cdot (x + e_x)|_p$. $|x|_p + e_y < p$ and $|2x|_p + e_y < p$. In this case $||x|_p + e_y|_{2^r} = |x|_p + e_y$, $||2x|_p + e_y|_{2^r} = |2x|_p + e_y$. For C_1 , an error (e_x, e_y) is missed if and only if

$$|x + e_x|_p = |x|_p + e_y. \quad (32)$$

Equivalently,

$$|e_x|_p = e_y. \quad (33)$$

For C_2 , an error (e_x, e_y) is missed if and only if

$$|2 \cdot (x + e_x)|_p = |2x|_p + e_y. \quad (34)$$

Equivalently,

$$|2e_x|_p = e_y. \quad (35)$$

From (33) and (35) we have $|e_x|_p = e_y = 0$. For a codeword x to mask a given error (e_x, e_y) , the following conditions must be satisfied:

$$x + e_x < 2^k, \quad (36)$$

$$|x|_p + e_y < p, \quad (37)$$

$$|2x|_p + e_y < p, \quad (38)$$

$$|e_x|_p = e_y = 0. \quad (39)$$

When (39) is satisfied, (37) and (38) are also satisfied. For each (e_x, e_y) such that $|e_x|_p = e_y = 0$, the total number of codewords in C_1 and C_2 that mask the error is $2 \cdot (2^k - e_x)$. For bad errors this number should be larger or equal to 2^k . Thus

$$2 \cdot (2^k - e_x) \geq 2^k. \quad (40)$$

Equivalently,

$$e_x \leq 2^{k-1}. \quad (41)$$

From (39) and (41), the number of bad errors is bounded by $\lceil \frac{2^{k-1} + 1}{p} \rceil$. Similarly, when C_1 and C_2 both belong to Case3, the number of bad errors is bounded by $\lceil \frac{2^{k-1}}{p} \rceil$.

So an upperbound of the total number of bad errors is

$$\left\lceil \frac{2^{k-1} + 1}{p} \right\rceil + \left\lceil \frac{2^{k-1}}{p} \right\rceil \approx 2 \cdot \left\lceil \frac{2^{k-1}}{p} \right\rceil. \quad (42)$$

The estimated probability of bad errors can be derived directly from (23). ■

All bad errors for $[|x|_p, |2x|_p]$ multilinear codes are in the format of $(e_x, e_y = 0)$. They have the same value as bad errors occurring to the information part of the linear arithmetic code. In another word, the proposed multilinear arithmetic code will not help if we assume that e_y is always 0. This situation, however, can be prevented by implementing a merged design of the original device and the encoder of the code since in this case faults will have high probability to affect not only the original device but also the encoder that generates the redundant part of the code. The advantage of this construction is that it has no other bad errors except for errors occurring to the information part of the code. The total number of bad errors is much smaller than that of linear arithmetic codes. Moreover, it is possible to reduce the number of bad errors to be nearly zero using $[|x|_p, |2x|_p]$ multilinear codes, which is impossible for linear arithmetic codes.

The next construction based on utilizing multiple modulus can drastically reduce the number of bad errors in the format of $(e_x, e_y = 0)$. The theorem can be proved in a similar way to the proof for Theorem 2. Here we omit the proof due to the limit of space.

Theorem 3. ($[p, q]$ *Multilinear Code*) Let C_1, C_2 be two arithmetic systematic codes defined by

$$C_i = \{(x, y) | x \in Z_{2^k}, y = f_i(x) \in Z_{2^r}\}, i \in \{1, 2\},$$

where $r = \max(\lceil \log_2 p \rceil, \lceil \log_2 q \rceil)$, p and q are different prime numbers larger than 2, $f_1(x) = |x|_p$ and $f_2(x) = |x|_q$ ($| \cdot |_p$ is the modulo operation). Without loss of generality, we assume that $p > q$. Denote by $e = (e_x \in Z_{2^k}, e_y \in Z_{2^r})$ the arithmetic additive errors and $\tilde{c} = c + e = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$ the distorted codeword. If we randomly select C_1 and C_2 with equal probability to encode the original messages and assume that the error only occurs to the information part of the code, the number of bad errors is upper bounded by

$$2 \cdot \left(\left\lceil \frac{2^{k-1}}{pq} \right\rceil + \left\lceil \frac{2^k}{pq} \right\rceil \right). \quad (43)$$

When $pq \ll 2^k$ and q is close to p , the estimated probability of bad errors occurring to the information part of $[p, q]$ multilinear codes is $3 \cdot 2^{-3r}$.

Remark 1. The precise number of bad errors for $[p, q]$ multilinear codes is hard to analyze. However, experimental results indicate that it is comparable to that of $[|x|_p, |2x|_p]$ multilinear codes and is much smaller than that of linear arithmetic codes. The idea of utilizing multiple residues as the redundant part of the code has already been presented in [7]. With two residues, the codeword was in the format of $(x, |x|_p, |x|_q)$. We want to emphasize that our construction is different from multi-residue codes proposed in [7] since at each clock cycle our code has only one residue for the redundant part. Instead of using multiple residues simultaneously, we use only one for each encoding and decoding operation and randomly select the modulus for different operations.

To demonstrate that $[p, q]$ multilinear codes have less bad errors in the format of $(e_x, e_y = 0)$ than linear and $[|x|_p, |2x|_p]$ multilinear arithmetic codes, we compare the number of bad errors in this class for the three constructions in Table 3. The number of information bits of the codes in the table is 32. For $[p, q]$ multilinear codes, q is selected to be the largest possible prime number less than p , e.g. when $p = 241$, $q = 239$. Linear arithmetic codes and $[|x|_p, |2x|_p]$ multilinear codes have the same number of bad errors occurring to the information part of the code. As p increases, this number for $[p, q]$ multilinear codes decreases much faster than the other two. When $p = 2767$ ($r = 12$), $[p, q]$ multilinear codes has only 1.7×10^3 bad errors in the format of $(e_x, 0)$ while the other two have 1.6×10^6 .

Table 3. Number of bad errors occurring to the information part of linear and multilinear codes ($k=32$)

	$p = 5$	$p = 241$	$p = 563$	$p = 883$	$p = 1237$	$p = 2767$
LinearArithmetic	8.6×10^8	1.8×10^7	7.6×10^6	4.9×10^6	3.5×10^6	1.6×10^6
$[x _p, 2x _p]$ codes	8.6×10^8	1.8×10^7	7.6×10^6	4.9×10^6	3.5×10^6	1.6×10^6
$[p, q]$ codes	8.6×10^8	2.2×10^5	4.1×10^4	1.7×10^4	8.5×10^3	1.7×10^3

To end this section, we summarize results on the probability of bad errors and the probability of bad errors occurring to the information part of linear and multilinear codes in Table 4. When bad errors occurring to the information part is more critical, $[p, q]$ multilinear codes should be used. Otherwise, $[|x|_p, |2x|_p]$ multilinear codes should be used because they are better than linear arithmetic codes in terms of the error detection abilities and require less hardware overhead to implement than $[p, q]$ multilinear arithmetic codes (Section 4).

Table 4. Probability of bad errors and probability of bad errors occurring to the information bits of linear and multilinear codes

Probability of	Linear Arithmetic Codes	$[x _p, 2x _p]$ codes	$[p, q]$ codes
bad errors	$\approx 0.3 \cdot 2^{-r+1}$	$\approx 2^{-2r}$	$\approx 2^{-2r^*}$
bad errors in info. part	$\approx 2^{-2r}$	$\approx 2^{-2r}$	$\approx 3 \cdot 2^{-3r}$

* : Based on experimental results.

4 Protection of Multipliers Using Multilinear Arithmetic Codes

In this section, we propose protection architectures for multipliers based on multilinear arithmetic codes. The multiplier is a basic block in many public key and even in some secret key cryptographic devices. Due to its arithmetic nature of the operations, arithmetic error model is most often used for such devices. In this section, we assume that additive arithmetic errors manifest themselves at the output of the multiplier and the predictor¹. The error is in the format of $e = (e_x, e_y)$, $e_x \in Z_{2^k}$, $e_y \in Z_{2^r}$, where

¹ The term *predictor* is used in this context to refer to the circuit that computes the checksum of the output of the operation directly from the inputs. In our case the predictor computes the checksum of the multiplication result.

k is the number of information bits and r is the number of redundant bits. We analyze and compare the number of bad errors for architectures protected by the three codes presented in Section 3. The advantage of architectures based on multilinear codes in terms of the number of bad errors is demonstrated.

The general hardware architecture of multipliers protected by block codes contains three parts: the original multiplier, the predictor that generates the redundant bits of the code and the error detection network which detects errors at the output of the device. The detailed architectures for the alternatives are shown in Figure 1.

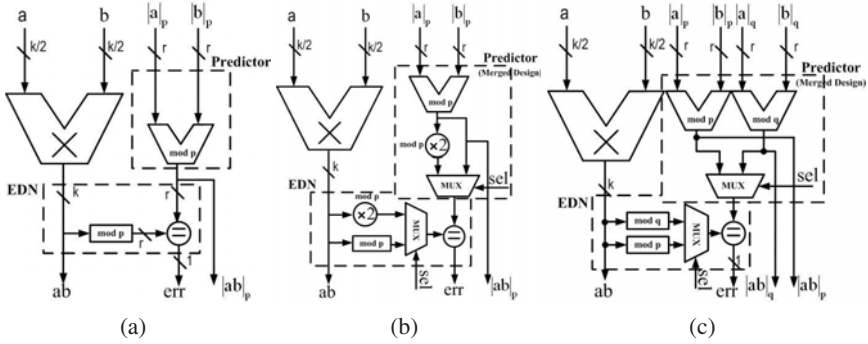


Fig. 1. Hardware architectures for multipliers protected by (a) linear arithmetic codes, (b) $[x_p, 2x_p]$ multilinear codes and (c) $[p, q]$ multilinear codes

The predictor for the linear arithmetic codes contains one multiplier in Z_p . Except for the r bit comparator, the only operation implemented in the error detection network (EDN) is a modulo p operation. The hardware overhead mainly comes from the $r = \lceil \log_2(p) \rceil$ bit multiplier, whose complexity is of the order of $O(r^2)$, and the modulo p operation in EDN, whose complexity is $O(k)$. (k is the number of information bits).

Compared with architectures based on linear arithmetic codes, the architecture utilizing $[x_p, 2x_p]$ multilinear codes only needs one extra r -bit multiplexer and one extra $\times 2$ operation in Z_p for both the predictor and the EDN. $\times 2$ operation is equal to shifting the operands by 1 bit, which is trivial in terms of the hardware overhead. The complexity of an r -bit multiplexer is in general of the order of $O(r)$. Thereby this architecture has comparable hardware overhead to linear arithmetic codes.

The protection architecture based on $[p, q]$ multilinear codes needs one more multiplier in Z_q for the predictor. When $p \ll 2^k$, which is often the case in real life, q should be selected as the largest prime number that is smaller than p if we want to minimize the number of bad errors. A multiplier in Z_q will have about the same hardware complexity as the multiplier in Z_p and this will double the overhead for the predictor. However, we claim that a merged design of the two multipliers for the predictor should be implemented. First, from the security point of view, separate redundant data path may be used by attackers to derive the secret information of the devices, e.g. the attacker can inject faults into one redundant path of the device which will never influence the other. A merged design can effectively solve the problem because most of the faults injected into the redundant part of the device will affect the generation of redundant bits

for both codes. Second, the hardware overhead of the predictor will be reduced if we merge the design of the two multipliers. A more aggressive approach is to design the original multiplier and the predictor of the code together as discussed in Section 3.

Remark 2. There is a tradeoff between the error detection abilities and the hardware overhead when we select p and q as specialized p and q can significantly reduce the hardware complexity of the modulo operation e.g. using Mersenne primes.

In Table 5 we compare the hardware overhead, the estimated probability of bad errors and the probability of bad errors occurring to the information part of the three architectures for the case when $k = 32, r = 7, p = 2^7 - 1$. For $[p, q]$ multilinear codes $q = 2^5 - 1$. The three designs were modeled in Verilog and synthesized in Synopsis Design Compiler. The hardware overhead for the predictor and EDN for linear arithmetic codes is in total 32%. With 7 redundant bits, the probability of bad errors for linear arithmetic codes is of the order of 2^{-7} . With similar hardware overhead, $[[x]_p, |2x|_p]$ multilinear code can reduce this probability to 2^{-14} . Experimental results shown that $[p, q]$ multilinear codes have nearly the same probability of bad errors as $[[x]_p, |2x|_p]$ multilinear codes, which is 2^{-14} . Meanwhile the probability of bad errors occurring to the information part is only about $3 \cdot 2^{-21}$, which is the smallest of the three. The disadvantage of this architecture is that it requires more overhead to implement. But we note that 53% is still less than the overhead to implement $(x, |x^2|_p)$ partially robust codes proposed in [10].

Table 5. Hardware overhead and the estimated probability of bad errors for architectures based on linear and multilinear codes ($k = 32, r = 7$)

	Linear Arith. Codes	$[[x]_p, 2x _p]$ ML codes	$[p, q]$ ML codes
Overhead for Predictors and EDN	32%	40%	53%
Probability of bad errors	$\approx 2^{-7}$	$\approx 2^{-14}$	$\approx 2^{-14}$ *
Probability of bad errors in the information part	$\approx 2^{-14}$	$\approx 2^{-14}$	$\approx 3 \cdot 2^{-21}$

* : Based on experimental results

We next present experimental results of the error detection capabilities of 8-bit multipliers protected by linear, $[[x]_p, |2x|_p]$, $[p, q]$ and $(x, |x^2|_p)$ arithmetic codes to demonstrate the advantages of architectures based on the proposed multilinear codes. All the four alternatives were simulated in MATLAB. The number of bad errors were analyzed and compared. In this experiment, $k = 16, p = 31$. For $[p, q]$ multilinear code $q = 29$. Each operand of the multiplier is a 8-bit binary vector and is randomly selected from 0 to $2^8 - 1$. Additive errors happen at the output of the multiplier and the predictors of the devices. The X axis in Figure 2 is the error masking probability. The Y axis shows the number of errors masked by a certain probability. Errors on the right half of each sub-figure are bad errors that are masked by a probability larger or equal to 0.5. As expected, the number of bad errors for (b),(c) and (d) are drastically reduced compared with architectures based on linear arithmetic codes. Compared with (d), the advantage of (b)

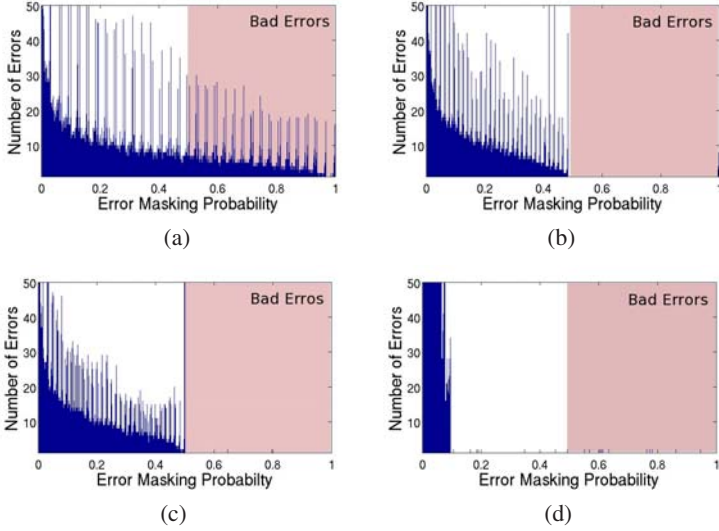


Fig. 2. Number of bad errors for 8-bit multipliers protected by (a) linear arithmetic codes, (b) $[|x|_p, |2x|_p]$ codes, (c) $[p, q]$ codes and (d) $(x, |x^2|_p)$ codes

and (c) is that they require less hardware overhead than the former. (For the estimation of the hardware overhead for $(x, |x^2|_p)$ code, please refer to [10]).

To further demonstrate the advantages of multilinear arithmetic codes, we compare the error/fault detection capabilities of the above four alternatives when errors at the output of the devices repeat. As discussed in Section 2, we assume that an attacker can use any fault injection methodologies and can inject faults with high spatial resolution to generate a specific error at the output of the device. Once injected, the fault/error stays for several clock cycles because the temporal resolution of the fault injection method is limited.

The X-Axis in Figure 3 is the number of consecutive clock cycles that the error lasts, which is denoted by t . Suppose the same error stays at the output for t consecutive clock cycles, the error is said to be detected if it is detected at least once among these t clock cycles. Otherwise, we say that the error is masked. Figure 3(a) compares the error masking probability for linear, $[|x|_p, |2x|_p]$, $[p, q]$ and $(x, |x^2|_p)$ arithmetic codes. The Y-Axis is the average error masking probability.

The average error masking probabilities of all four alternatives decrease as t increases. However, $[|x|_p, |2x|_p]$, $[p, q]$ and $(x, |x^2|_p)$ arithmetic codes have much better error detection capabilities than linear codes for large t . Figure 3(b) plots the ratio of the average error masking probability of $[|x|_p, |2x|_p]$, $[p, q]$ and $(x, |x^2|_p)$ codes to that of the linear arithmetic codes. When the error stays for two consecutive clock cycles, the error detection abilities for multilinear and robust arithmetic codes are already twice better than that of linear codes. $(x, |x^2|_p)$ has the lowest error masking probability when $t = \{2, 3, 4\}$. For $t > 6$, $[|x|_p, |2x|_p]$ codes and $(x, |x^2|_p)$ codes have similar performance. $[p, q]$ codes are the best among the three nonlinear arithmetic codes when $t > 4$ and $p = 31, q = 29$. For smaller q , the error detection ability of $[p, q]$ codes will be

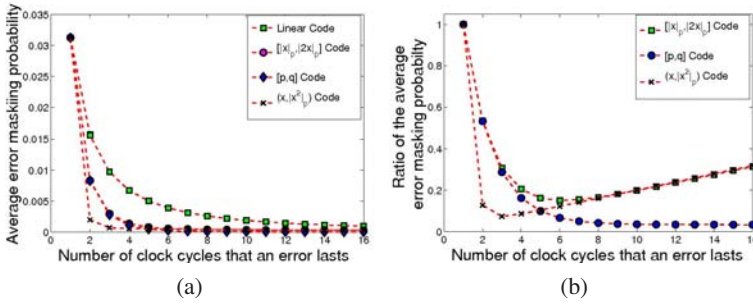


Fig. 3. Error detection properties of 8-bit multipliers protected by linear, multilinear and partially robust arithmetic codes in lazy channels

worse but still better than linear arithmetic codes. Thereby, the proposed multilinear arithmetic codes can provide comparable or even better error protection abilities than partially robust arithmetic codes $(x, |x^2|_p)$ while requiring much less hardware overhead. Architectures based on nonlinear arithmetic codes are better than that based on linear arithmetic codes in terms of the number of bad errors and have lower error masking probability in lazy channels where errors tend to repeat.

5 Conclusions

In this paper we presented several constructions of multilinear arithmetic codes and compared their error detection properties to that of linear and partially robust arithmetic codes. Architectures of reliable multiplier based on multilinear arithmetic codes were introduced. Experimental results show that the error detection capability can be significantly improved over that of linear codes at the expense of a very mild increase in the hardware overhead (Table 5). The proposed multilinear codes can achieve as good error detection abilities as that of the partially robust arithmetic codes with smaller hardware overhead and are better choices than linear arithmetic codes in lazy channels where errors tend to repeat. These codes can efficiently prevent the attackers from injecting undetectable faults/errors under the assumption that the temporal resolution of the fault injection methodologies is limited. Finally, we would like to point out that in this paper we considered only the case when multilinear codes are constructed from 2 codes. But the results can be easily generalized to any number of codes.

References

- [1] Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
- [2] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
- [3] Skorobogatov, S.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003)

- [4] Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerers apprentice guide to fault attacks (2002)
- [5] Boneh, D., Demillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology* (2001)
- [6] Piret, G., Quisquater, J.-J.: A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
- [7] Rao, T., Garcia, O.: Cyclic and multiresidue codes for arithmetic operations. *IEEE Transactions on Information Theory* IT-17(1) (1971)
- [8] Karpovsky, M., Kulikowski, K., Taubin, A.: Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In: Proc. Int. Conference on Dependable Systems and Networks (DSN) (July 2004)
- [9] Gaubatz, G., Sunar, B., Karpovsky, M.G.: Non-linear residue codes for robust public-key arithmetic. In: Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) (2006)
- [10] Kulikowski, K., Wang, Z., Karpovsky, M.G.: Comparative analysis of fault attack resistant architectures for private and public key cryptosystems. In: Proc of Int. Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC (2008)
- [11] Carlet, C., Ding, C.: Highly nonlinear mappings. *Journal of Complexity* 20(2-3) (2004)
- [12] Vasylysov, I., Hambardzumyan, E., Kim, Y.-S., Karpinsky, B.: Fast digital TRNG based on metastable ring oscillator. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 164–180. Springer, Heidelberg (2008)
- [13] Wang, Z., Karpovsky, M., Sunar, B.: Multilinear codes for robust error detection. In: IEEE International On-Line Testing Symposium, IOLTS (2009)

Hardware/Software Co-design of Public-Key Cryptography for SSL Protocol Execution in Embedded Systems

Manuel Koschuch¹, Johann Großschädl², Dan Page², Philipp Grabher²,
Matthias Hudler¹, and Michael Krüger¹

¹ FH Campus Wien – University of Applied Sciences,
Favoritenstraße 226, A-1100 Vienna, Austria

{manuel.koschuch,matthias.hudler,michael.krueger}@fh-campuswien.ac.at

² University of Bristol, Department of Computer Science,
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
{johann,page,grabher}@cs.bris.ac.uk

Abstract. Modern mobile devices like cell phones or PDAs allow for a level of network connectivity similar to that of standard PCs, making access to the Internet possible from anywhere at anytime. Going along with this evolution is an increasing demand for cryptographically secure network connections with such resource-restricted devices. The Secure Sockets Layer (SSL) protocol is the current de-facto standard for secure communication over an insecure network like the Internet and provides protection against eavesdropping, message forgery and replay attacks. To achieve this, the SSL protocol employs a set of computation-intensive cryptographic algorithms, in particular public-key algorithms, which can result in unacceptably long delays on devices with modest processing capabilities. In this paper we introduce a hardware/software co-design approach for accelerating SSL protocol execution in resource-restricted devices. The software part of our co-design consists of MatrixSSLTM, a lightweight SSL implementation into which we integrated elliptic curve cryptography (ECC) to speed up the public-key operations performed during the SSL handshake. The hardware part comprises a SPARC V8 compliant processor core with instruction set extensions to support the low-level arithmetic operations carried out in ECC. Our co-design executes a full SSL handshake using an elliptic curve over a 192-bit prime field in less than 300 msec when the SPARC processor is clocked at 20 MHz. A pure software implementation like OpenSSL is, depending on the field type and order, up to a factor of 10 slower than our co-design solution.

1 Introduction

The current de-facto standard for secure communication over an insecure, open medium like the Internet is the Secure Sockets Layer (SSL) protocol [9] and its successor, the Transport Layer Security (TLS) protocol [8,33]. Both use a combination of public-key and secret-key cryptographic techniques to ensure confidentiality, integrity and authenticity of communication between two parties

(typically referred to as client and server). The SSL protocol is composed of two layers and includes several sub-protocols. At the lower level is the SSL Record Protocol, which specifies the format used to transmit data between client and server (including encryption and integrity checking) [9]. It encapsulates various higher-level protocols, one of which is the SSL Handshake Protocol. The main tasks of the handshake protocol are the negotiation of a set of cryptographic algorithms, the authentication of the server (and, optionally, of the client¹), as well as the establishment of a *pre-master secret* via asymmetric (i.e. public-key) techniques [9]. Both the client and the server derive a master secret from this pre-master secret, which is then used by the record protocol to generate shared keys for symmetric encryption and message authentication [9]. The handshake protocol, on the other hand, relies on services provided by the record protocol to exchange messages between client and server.

The SSL/TLS protocol is algorithm-independent (or algorithm-agile) in the sense that it supports different algorithms for one and the same cryptographic operation, and allows the communicating parties to make a choice among them [9]. At the beginning of the handshake phase, the client and the server negotiate a *cipher suite*, which is a well-defined set of algorithms for authentication, key agreement, symmetric encryption, and integrity checking. Both SSL and TLS specify the use of RSA or DSA for authentication, and RSA or Diffie-Hellman for key exchange. In 2006, the TLS protocol was revised to include ECDSA as signature primitive and ECDH for key exchange [5]. The big benefit of Elliptic Curve Cryptography (ECC) [19] over traditional public-key schemes operating in \mathbb{Z}_n or \mathbb{Z}_p^* is its better security-per-bit ratio: A carefully chosen 160-bit ECC cryptosystem attains a security level comparable to that of 1024-bit RSA. As a consequence, public-key schemes based on elliptic curves over finite fields can use significantly shorter keys compared to their “classical” counterparts. These reduced key lengths translate directly into memory and bandwidth savings when SSL handshakes are performed with one of the ECC-based cipher suites from [5] instead of an RSA cipher suite. In addition, certain cryptographic operations (e.g. generation of signatures, key exchange) can be executed much faster in an elliptic curve group than in a multiplicative group like \mathbb{Z}_p^* .

The advent of the wireless Internet has created a strong demand for secure communication via mobile devices such as cell phones or PDAs. However, these devices are battery-operated, and hence severely constrained by computational resources (processing power, memory, network bandwidth, etc.). When implementing SSL for mobile devices, great care must be taken to utilize the scarce resources as efficiently as possible [2,3,14]. The delay a user experiences when establishing an SSL connection depends heavily on the execution time of the public-key operations carried out during the handshake (i.e. authentication and key agreement). If an RSA-based cipher suite is used, the client has to perform

¹ Most Internet applications use SSL only for server-side authentication, which means that the server is authenticated to the client, but not vice versa. Client authentication is typically done at the application layer (and not the SSL layer), e.g. by entering a password and sending it to the server over a secure SSL connection.

two modular exponentiations, one for the verification of the server's certificate and one for the encryption of the pre-master secret². Even though these exponentiations involve public exponents (which are usually small), they constitute a significant overhead. For example, Gupta et al [14] analyzed the performance of an SSL client written in Java on a 20 MHz Palm Vx and found that the two RSA public-key operations account for almost 30% of the overall execution time of the SSL handshake³. On the other hand, when using an ECC cipher suite, the public-key operations (i.e. ECDSA verification, ECDH key exchange) make up more than 80% of the handshake time⁴ [23]. Therefore, hardware acceleration of the public-key operations carried out during the handshake is desirable.

The straightforward approach to hardware acceleration of public-key cryptography is the integration of a dedicated co-processor to off-load the computationally expensive parts of an algorithm (e.g. modular exponentiation in the case of RSA, scalar multiplication in ECC) from the main processor [7,17]. In the embedded realm, however, fixed-function hardware accelerators in the form of cryptographic co-processors exhibit a number of disadvantages. Co-processors for RSA generally demand large silicon area, which poses a particular problem for low-cost embedded devices. On the other hand, co-processors for ECC often lack the flexibility to support the multitude of implementation options that are recommended by several standardization organizations around the world. One example of these options is the large number of "standardized" finite fields upon which elliptic curve cryptosystems can be built [34]. Supporting various fields of different characteristic and order is difficult with a fixed-function (i.e. hard-wired) accelerator and may also consume a large amount of silicon area. Given the algorithm-agile nature of the SSL protocol, it seems questionable whether a cryptographic co-processor can meet the desired level of flexibility at moderate hardware cost. Modern security protocols, such as SSL or IPsec, are constantly evolving and hence changing their repertoire of crypto algorithms (e.g. to phase out compromised algorithms, to include new algorithms, or to adapt the minimal key size of algorithms), which again calls for a flexible and scalable approach to hardware acceleration.

In this paper we present a new methodology for hardware acceleration of the SSL handshake based on *hardware/software co-design* [35] of the involved cryptographic algorithms. The specific co-design approach we followed in our work is the integration of custom instructions into a general-purpose processor to speed up the processing of performance-critical arithmetic operations carried out in ECC (e.g. multiplication in finite fields). Hardware/software co-design at the

² Instead of sending a single certificate to the client, the server may also send a chain of two or more certificates linking the server's certificate to a trusted certification authority (CA). However, throughout this paper we assume that the certificate chain consists of just one certificate, and hence a single signature verification operation is sufficient to check the validity of the certificate.

³ A 1024-bit modular exponentiation with a public exponent of 65537 executes in 1433 msec [14, Table I], and the full SSL handshake takes approximately 10 seconds.

⁴ We will argue in Subsection 3.2 why ECC is advantageous over RSA for client-side SSL processing on resource-constrained devices.

granularity of instruction set extensions is particularly area-efficient and allows one to retain the full flexibility of a “pure” software solution, which makes this approach perfectly well suited for hardware acceleration of the SSL protocol in low-cost embedded systems. The hardware part of our co-design comprises an embedded SPARC V8 processor into which we integrated a set of six custom instructions to facilitate the efficient execution of arithmetic operations in prime and binary fields of large order [13]. The software part consists of MatrixSSL, a “lightweight” SSL implementation written in ANSI C [29]. MatrixSSL provides both client and server functionality, but lacks support for ECC. Therefore, we developed a simple crypto library including RSA, DSA, Diffie-Hellman, as well as ECC over prime and binary fields, and integrated it into MatrixSSL along with the ECC cipher suites from [5]. Our experimental results show that, due to the lightweight implementation of the SSL stack, the speed-up gained at the low-level field arithmetic propagates almost lossless up to the application layer. We also compare the results of our co-design with the performance figures of a pure software implementation of the SSL protocol, namely OpenSSL [28]. This comparison confirms that hardware/software co-design in the form of instruction set extensions for public-key cryptography, in particular ECC, is a good way to accelerate the SSL handshake.

2 Public-Key Cryptography

The SSL/TLS protocol makes heavy use of public-key cryptography during the handshake phase to accomplish such tasks as authentication and key establishment. In this section we briefly discuss implementation aspects of both classical public-key cryptosystems (RSA, DSA, Diffie-Hellman) as well as elliptic curve cryptosystems in the context of the SSL handshake.

2.1 RSA, DSA, Diffie-Hellman

The RSA cryptosystem operates in the residue class ring \mathbb{Z}_n , where n is the product of two large primes. DSA and Diffie-Hellman, on the other hand, use the multiplicative group \mathbb{Z}_p^* (or a subgroup thereof) as underlying algebraic structure. The basic operation of all these cryptosystems is exponentiation, i.e. the repeated application of the ring or group operation, namely multiplication, to an element of the ring (resp. group). Of course, the multiplications are performed modulo n (or modulo p , respectively), which means that said exponentiation is actually a modular exponentiation of the form $c = m^e \bmod n$ [24]. In case of the RSA algorithm, the modulus n is a product of primes, the exponent e satisfies $\gcd(e, \phi(n)) = 1$, and the base m is in the interval $[0, n - 1]$, i.e. $m \in \mathbb{Z}_n$. The security of the RSA cryptosystem is closely related to the Integer Factorization Problem (IFP), even though no mathematical proof exists that the factorization of n is needed to break RSA. Factoring an RSA modulus is widely believed to be computationally infeasible if its prime factors are large (e.g. ≥ 512 bits). On the other hand, the security of DSA and Diffie-Hellman relies on the Discrete

Logarithm Problem (DLP) in \mathbb{Z}_p^* , which is defined as follows: Given a generator g for \mathbb{Z}_p^* (or a subgroup thereof) and an element a of said (sub)group, find the integer x such that $a = g^x \bmod p$. The DLP is considered intractable, provided that the group \mathbb{Z}_p^* and the generator g are properly chosen.

The standard algorithm for computing a modular exponentiation $m^e \bmod n$ is the *square-and-multiply algorithm*, which is also referred to as binary exponentiation method [24] since it uses the binary expansion of the exponent e . Two variants of the binary method are described in [24]; one scans the bits of e from left to right (i.e. MSB first), the other from right to left (i.e. LSB first). Assuming an exponent e of length $l = 1 + \lceil \log_2 e \rceil$ bits, the square-and-multiply algorithm executes l modular squarings and roughly $l/2$ modular multiplications, with the exact number depending on the Hamming weight of e . The number of modular multiplications can be reduced if some extra memory for storing powers of the base m is available. For example, the *k-ary exponentiation method* (also called window method) processes k bits of the exponent e at a time, thereby reducing the number of modular multiplications to l/k in the worst case. However, the *k-ary exponentiation* requires pre-computation and storage of 2^k powers of the base m (see Algorithm 14.82 in [24]), which is why this method is rarely implemented on resource-constrained embedded devices like smart cards. If the base m is fixed and known a-priori (which is, for example, the case when generating a DSA signature), the number of both modular multiplications and squarings can be reduced through the *fixed-based comb method* as described in [24].

The execution time of a modular exponentiation depends heavily on the implementation of the two operations it consists of, namely modular multiplication and modular squaring. Both operations include a modular reduction, which can be efficiently performed using the well-known Montgomery technique [25]. Koç et al [22] describe several optimized software algorithms for Montgomery multiplication, among these is the so-called Coarsely Integrated Operand Scanning (CIOS) method. The CIOS method executes a total of $2s^2 + s$ single-precision (i.e. w -bit) multiply instructions, whereby n denotes the number of w -bit words that are needed to accommodate an n -bit operand, i.e. $s = \lceil n/w \rceil$ (see [22] for a detailed analysis).

The computational cost of a modular exponentiation can be reduced if the exponent e and/or the base m are suitably chosen, which is possible for RSA as well as DSA and Diffie-Hellman. For example, it is common practice to choose a small public exponent in RSA; a typical value is $2^{16} + 1$. In this case, operations involving the public exponent (e.g. RSA encryption) are significantly faster than operations involving the private exponent, even when the latter are supported by the Chinese Remainder Theorem [24]. On the other hand, Diffie-Hellman and DSA can use special primes to simplify the reduction operation. In addition, the generator g used in Diffie-Hellman key exchange can be small (e.g. $g = 2$), which reduces the cost of a modular exponentiation. DSA implementations generally take advantage of a generator g that generates a (large) subgroup of \mathbb{Z}_p^* , e.g. a 160-bit subgroup when p is a 1024-bit prime, which considerably alleviates the computational burden of a modular exponentiation with g as base.

2.2 Elliptic Curve Cryptography

Elliptic curve cryptosystems operate in an additive Abelian group, namely the group of points on an elliptic curve defined over a finite field. A discussion of the mathematical foundations of ECC is beyond the scope of this paper; we refer the interested reader to textbooks such as [19] and [4]. In short, ECDSA and ECDH can be seen as the elliptic-curve “equivalents” of the classical DSA and Diffie-Hellman cryptosystems, whereby the group \mathbb{Z}_p^* is replaced by $E(\mathbb{F}_q)$, the group of \mathbb{F}_q -rational points on a curve E . The basic building block of all ECC schemes is *scalar multiplication*, i.e. an operation of the form $k \cdot P$ where P is a point on the curve (i.e. $P \in E(\mathbb{F}_q)$) and k is an integer [4]. Scalar multiplication in an additive group corresponds to exponentiation in a multiplicative group; both are performed through repeated application of the group operation to an element. However, the group operation in $E(\mathbb{F}_q)$ is the addition of points, which in turn is realized by a sequence of arithmetic operations in the underlying finite field \mathbb{F}_q . The security of both ECDH and ECDSA is based on the intractability of the elliptic curve discrete logarithm problem (ECDLP), which can be defined as follows: Given an elliptic curve group $E(\mathbb{F}_q)$, a base point $P \in E(\mathbb{F}_q)$, and a second point $Q \in E(\mathbb{F}_q)$, find the smallest integer k so that $Q = k \cdot P$, provided such an integer exists. Currently, the fastest algorithm known for solving the ECDLP requires fully exponential time when $E(\mathbb{F}_q)$ and the base point P were chosen with care. As a consequence, ECC schemes can use much shorter keys than their “classical” counterparts based on the DLP or the IFP (e.g. 160 bits instead of 1024 bits) [19].

Ephemeral ECDH key exchange requires the server and the client to execute two scalar multiplications of the form $k \cdot P$; one to generate a key pair and the other to obtain the shared secret key. On the other hand, the generation of an ECDSA signature costs just one scalar multiplication $k \cdot P$, but the verifier has to execute a double scalar multiplication of the form $k \cdot P + l \cdot Q$ [4]. Similar to the square-and-multiply algorithm for exponentiation, a scalar multiplication can be carried out via *point additions* and *point doublings*, both of which, in turn, involve a sequence of arithmetic operations (i.e. addition, multiplication and inversion) in the underlying finite field \mathbb{F}_q [19]. Inversion is by far the most expensive field operation. However, it is possible to add points on an elliptic curve without the need to perform costly inversions, e.g. by representing the points in *projective coordinates* [19]. When using projective coordinates, an entire scalar multiplication can be carried out solely with field additions (resp. subtractions) and field multiplications (resp. squarings); just a single inversion is necessary to convert the result from projective coordinates back to the conventional (affine) coordinate system.

Before an ECDH key exchange (or any other elliptic curve scheme) can be carried out, the involved entities have to agree upon a common set of so-called *domain parameters* [19], which specify the field \mathbb{F}_q , the elliptic curve E (i.e. the coefficients $a, b \in \mathbb{F}_q$ defining the curve), a base point $P \in E(\mathbb{F}_q)$ generating a cyclic subgroup of large order, the order n of this subgroup, and the co-factor

$h = \#E(\mathbb{F}_q)/n$. Consequently, elliptic curve domain parameters are simply a sextuple $D = (q, a, b, P, n, h)$.

The efficient implementation of the field arithmetic, in particular the multiplication, has a major impact on the performance of ECC cryptosystems. Prime fields and binary extensions fields are especially important since they have been recommended by numerous standards bodies around the world. The elements of a prime field \mathbb{F}_p are simply the integers $0, 1, 2, \dots, p-1$, and the arithmetic operations are addition and multiplication modulo p . Therefore, all algorithms for arithmetic in \mathbb{Z}_p^* can be used for \mathbb{F}_p as well, e.g. Montgomery reduction as described in [22]. However, it is possible (and common practice) to use special primes in ECC for which optimized modular reduction methods exist; a typical example are the generalized-Mersenne (PM) primes that are specified in some standards, e.g. in [34]. For example, the reduction of a 384-bit integer modulo the 192-bit GM prime $p = 2^{192} - 2^{64} - 1$ can be performed with three simple 192-bit additions. GM primes allow one to achieve better performance with the trade-off that each GM-prime requires a different reduction routine, resulting in large code size if all standardized GM-primes are to be supported.

The elements of a binary finite field \mathbb{F}_{2^m} are binary polynomials of degree up to $m-1$; the arithmetic in \mathbb{F}_{2^m} is polynomial arithmetic (i.e. addition and multiplication of binary polynomials) performed modulo an irreducible polynomial $p(t)$ of degree m . Addition in \mathbb{F}_{2^m} is equivalent to exclusive-or and can be realized using the processor's XOR instruction on words of the operands. The major disadvantage of binary fields is that multiplication is relatively costly in software. Multiplication in \mathbb{F}_{2^m} consists of a polynomial multiplication over \mathbb{F}_2 , followed by a reduction of the product modulo the irreducible polynomial. The former is typically realized with the basic Shift-and-XOR method or one of its optimized variants such as the left-to-right comb method (see Algorithm 2.36 in [19]). Combining this method with Karatsuba's technique can be advantageous for large operands [21]. Also the reduction modulo $p(t)$ requires just shift and XOR instructions, and is relatively fast when $p(t)$ is sparse. Squaring in \mathbb{F}_{2^m} is a linear operation and, hence, requires just a fraction of the execution time of a multiplication.

3 Secure Sockets Layer (SSL) Protocol

The Secure Sockets Layer (SSL) protocol and its successor, the Transport Layer Security (TLS) protocol, are standardized protocol suites for enabling secure communication between a client and a server over an insecure network [8]. The main focus in the design of these protocols lay in modularity, extensibility, and transparency. Both SSL and TLS use a combination of asymmetric (i.e. public-key) and symmetric (i.e. secret-key) cryptographic techniques to authenticate the communicating parties and encrypt the data being transferred. The actual algorithms to be used for authentication and encryption are negotiated during the handshake phase of the protocol. SSL/TLS supports traditional public-key cryptosystems (i.e. RSA, DSA, Diffie-Hellman) as well as elliptic curve systems such as ECDSA and ECDH.

Table 1. SSL handshake, optional messages printed *italic*

Client	Server
ClientHello	
	ServerHello
	<i>Certificate</i>
	<i>ServerKeyExchange</i>
	<i>CertificateRequest</i>
	ServerHelloDone
<i>Certificate</i>	
ClientKeyExchange	
<i>CertificateVerify</i>	
ChangeCipherSpec	
Finished	
	ChangeCipherSpec
	Finished
Application Data	Application Data

3.1 SSL Handshake

The SSL protocol contains several sub-protocols, one of which is the handshake protocol. After agreeing upon a *cipher suite*⁵ that defines the cryptographic primitives to be used and their domain parameters, the server (and possible the client too) is authenticated and a pre-master secret is established using public-key techniques. Table 1 shows an overview of this process (see 9 for a more detailed description). When using an RSA cipher suite, the pre-master secret is established through key transport: The client generates a random number and sends it in RSA-encrypted form to the server. On the other hand, when using an ECC-based cipher suite, the pre-master secret is established through a key exchange to which both the client and the server contribute randomness.

In the *ClientHello* message the client sends its supported cipher suites to the server, who confirms the selected suite in its own *ServerHello* message. Then, the server transmits its certificate and an optional request for authentication to the client. In most cases there is no mutual authentication and only the server presents its certificate to the client. The client is rarely authenticated during the handshake phase, but rather thereafter, e.g. by sending a password to the server. The client then verifies the server's certificate and answers with the *ClientKeyExchange* message, containing the material needed for the server to derive the shared pre-master secret. If the public key extracted from the server's certificate can not be used for encryption (e.g. because it is only authorized to signing), then the server sends a *ServerKeyExchange* message including a second public key. The *ChangeCipherSpec* is just a status message, telling both parties to use the negotiated suite from now on. The final *Finished* message is then the first one encrypted with the selected cipher and the symmetric key, derived from the pre-master secret.

⁵ A cipher suite is a pre-defined combination of three cryptographic algorithms: A key exchange/authentication algorithm, an encryption algorithm, and a MAC algorithm.

The expensive steps in this process are the verification of the certificate's signature (using, for example, RSA, DSA, or its elliptic curve equivalent ECDSA) and the establishment of the shared pre-master secret, which is usually done in one of two ways, depending on the cipher suite chosen.

- If RSA is chosen, the client creates a random value, encrypts it with the server's public key (one modulo exponentiation) and sends the result back to the server, who can decrypt it (another modulo exponentiation).
- If ECDH is chosen, the client creates a random value k , calculates $R = k \cdot P$ (first scalar multiplication) and sends this value to the server. Then it calculates $k \cdot Q$ (second scalar multiplication), with Q being the server's public key, obtained from its certificate. The server finally performs a single scalar multiplication using R and its own private key. The final result for both server and client is the shared pre-master secret (see [5] for details).

3.2 Advantages of ECC Cipher Suites over RSA Cipher Suites

When an RSA cipher suite is used for the handshake, the client has to perform two modular exponentiations: one to verify the RSA signature contained in the server's certificate, and the other to encrypt the pre-master secret. Both of these exponentiations are carried out with public exponents, which are usually small [18,31]. Unfortunately, when using an ECC-based cipher suite, the situation is less favorable for the client. ECDH key exchange requires the client to execute two scalar multiplications, while the verification of an ECDSA signature involves a double-scalar multiplication of the form $k \cdot P + l \cdot Q$ [19]. ECDSA signature verification is the most costly of the cryptographic operations performed during an ECC-based handshake. In summary, RSA cipher suites impose high computational load on the server but low overhead on the client [1], while in general the opposite holds when an ECC cipher suite is used [16]. It is widely believed that RSA cipher suites are to prefer over their ECC-based counterparts when the SSL client runs on an embedded device with modest resources, while ECC cipher suites yield considerable better performance (i.e. throughput) figures on the server side [15,30]. However, there exist also numerous good arguments in favor of using ECC-based cipher suites on resource-restricted clients:

- Elliptic curve cryptosystems can use much shorter keys than RSA schemes to ensure a certain level of security (e.g. 160 vs. 1024 bits), which translates directly into memory and bandwidth savings. The former is important for low-cost devices with small memory, while the latter is relevant for mobile and battery-powered devices since wireless data transmission is very costly in terms of energy [32].
- Results from the literature confirm that a 1024-bit RSA signature can be verified significantly faster than a 160-bit ECDSA signature. However, the picture changes with higher security levels (i.e. longer keys). Brown et al [6] found that when using a Koblitz curve over $\mathbb{F}_{2^{233}}$, an ECDSA signature can be verified in 5,878 msec on a Palm V device, whereas the verification of an

RSA signature of roughly comparable strength (i.e. 2048-bit modulus and 17-bit public exponent) takes 7,973 msec.

- The key length of ECC scales linearly with that of symmetric ciphers such as the AES. For example, the NIST [26] recommends to use 128-bit AES in combination with 256-bit ECC or 3,072-bit RSA. However, 256-bit AES demands RSA keys with a length of 15,360 bits for equivalent security, while 512-bit keys suffice when using ECC. This linear scaling property makes a good case for ECC if AES-equivalent security levels are to be supported.
- Even though this paper focusses on client-side acceleration of SSL, it should be noted that ECC offers significant performance-advantages for SSL servers [15][16]. RSA cipher suites are highly computation-intensive on the server side [7][36], which may also impact the overall latency of the handshake, in particular if the server is under heavy load.
- When performing an SSL handshake with client authentication, ECC-based cipher suites are, in general, less costly than their RSA counterparts (this applies to both sides, the server *and* the client [15][30]).
- Using a cipher suite with ephemeral ECDH key exchange provides forward secrecy, whereas RSA-based key transport does not [4]. Another advantage of ECDH key exchange is that both the client and the server can contribute randomness to the generation of the pre-master secret, which is not the case with RSA-based key transport.
- ECDSA, ECDH, and ECMQV (an authenticated variant of ECDH) are the only public-key schemes included in NSA Suite B [27], i.e. RSA must not be used to secure sensitive or classified U.S. government communications.

For all these reasons we decided to use ECC cipher suites for the performance evaluation of our co-designed SSL stack. However, other cipher suites based on RSA, DSA, or Diffie-Hellman are also supported.

4 Implementation Details and Results

The hardware platform we used for our co-design is SPARC V8 softcore into which we integrated a small set of custom instructions to speed up public-key cryptography. Instruction set extension is a simple and efficient way to enhance a processor’s capabilities to support special application domains. In contrast to a dedicated co-processor, the hardware overhead of custom instructions is, in general, relatively small. Moreover, since the instructions are directly integrated into the ordinary processing pipeline, there is no need for expensive operand transfers, which can heavily affect the performance of such solutions. Dedicated co-processors are also limited in terms of flexibility and usually not designed to support such a multitude of cryptographic algorithms as is needed in SSL.

Software implementations of cryptosystems often spend the majority of execution time in a few performance-critical code sections (e.g. inner loops), which makes it amenable to processor customization. It was shown in [12] that a small set of only five or six custom instructions suffices to accelerate the full domain

Table 2. Format and description of the CIS instructions for public-key cryptography

Format	Description	Operation
<code>umac rs1, rs2</code>	Unsigned Multiply & Accumulate	$accu \leftarrow accu + rs1 \times rs2$
<code>umac2 rs1, rs2</code>	Unsigned Mul. & Accumulate Twice	$accu \leftarrow accu + 2(rs1 \times rs2)$
<code>uaddac rs1, rs2</code>	Add to Accumulator Unsigned	$accu \leftarrow accu + rs1 + rs2$
<code>shacr rd</code>	Shift Accu Registers Right	$rd \leftarrow accu[31:0]; accu \leftarrow accu \gg 32$
<code>gf2mul rs1, rs2</code>	Bin. Polynomial Multiply	$accu \leftarrow rs1 \otimes rs2$
<code>gf2mac rs1, rs2</code>	Bin. Polynomial Mul. & Accumulate	$accu \leftarrow accu \oplus rs1 \otimes rs2$

of public-key primitives specified in the IEEE Standard 1363 [20]; these include traditional cryptosystems such as RSA, but also ECC systems over both prime fields and binary extension fields. Based on the ideas in [11,12], we devised the *Cryptography Instruction Set (CIS) extensions* to the SPARC V8 architecture [13] and integrated them into the LEON-2 softcore, an open-source SPARC V8 implementation developed by Gaisler Research. The LEON-2 VHDL model can be synthesized to FPGA and standard cell technologies [10].

The CIS extensions for PKC consist of a set of six custom instructions (see Table 2) and a functional unit (FU) on which the instructions are executed. This FU is basically a multiply-accumulate (MAC) unit composed of a (32×16) -bit unified multiplier and a 72-bit accumulator. A so-called unified multiplier is a multiplier that uses the same datapath for two different types of operands, namely integers and binary polynomials [12]. The MAC unit also contains three result accumulation registers (`%asr20`, `%y`, and `%asr18`), in the following called *accu registers*. Besides the six custom instructions shown in Table 2, the MAC unit is capable to execute the two “native” SPARC V8 multiply instructions `umul` and `smul`. Consequently, the CIS extensions can be easily integrated into a SPARC V8 core by replacing the original integer multiplier by a MAC unit for integers and binary polynomials and modifying the instruction decoder to support the custom instructions.

Most of the CIS instructions listed in Table 2 get two 32-bit words from the general-purpose register file as input and place the result in the accu registers. The `umac` instruction can be used to implement the inner loop of long integer multiplication according to the product scanning technique and is also useful for Montgomery multiplication [11]. Long integer squaring can be efficiently executed with help of the `umac2` instruction. The two instructions `uaddac` and `shacr` facilitate the modular reduction operation for the special primes used in EC cryptography. Finally, the instructions `gf2mul` and `gf2mac` interpret their operands as binary polynomials and perform polynomial multiply/MAC operations that allow to speed up EC systems based on binary fields. A detailed description of the custom instructions and their use in the diverse arithmetic algorithms can be found in [11,12].

Especially for binary extension fields, the presence of hardware support for polynomial multiplication offers a significant performance gain compared to a native software implementation. The additional instructions are easily accessible through a modified assembler and the use of inline assembly in ordinary

C programs. Due to their generic nature, they can be used for all sorts of cryptographic algorithms requiring fast integer or polynomial arithmetic.

We integrated the CIS extensions into the LEON-2 core and prototyped the extended processor in an FPGA. The CIS extensions have no impact on the cycle time, i.e. the extended LEON-2 can be clocked with the same frequency as the “original” LEON-2 processor (up to 50 MHz in our FPGA device). We also synthesized the extended LEON-2 using a 0.35μ standard cell library and found that the CIS extensions entail an increase in area by merely 5,550 gates compared to a baseline LEON-2 core with a (32×16) -bit multiplier.

4.1 Evaluation of Code Size and Performance

The software part of our co-designed SSL stack is based on the freely available MatrixSSL library [29]. MatrixSSL in its original form provides both client and server functionality, but does not feature ECC. Therefore, we developed a lightweight public-key cryptographic library and integrated it into MatrixSSL so as to support the ECC cipher suites specified in [5]. We used OpenSSL [28] as a reference implementation with respect to code size and performance. Similar to OpenSSL, our implementation is generic in the sense that it works for every curve over prime or binary extension fields and allows free combination of the cryptographic primitives (e.g. using ECDSA as signature primitive and RSA for key establishment). Table 3 shows a comparison between our implementation (i.e. MatrixSSL+ECC), the original MatrixSSL version (without ECC support) and the OpenSSL library in terms of source files and code size. The integration of ECC increased the size of MatrixSSL by just 15-20%. For comparison, the OpenSSL executable is almost 20 times larger.

The crypto library we integrated into MatrixSSL is realized in a very straightforward way. We used Algorithm 2.9 in [19] to implement the multiple-precision multiplication and Montgomery’s well-known algorithm for modular reduction [25]. In order to keep the size of our library at a minimum, we did not include optimized reduction functions for special primes like the NIST primes. Also the curve arithmetic over \mathbb{F}_p is based on well-known algorithms. We represent the elliptic curve points using the mixed Jacobian-affine coordinates described in [19, Section 3.2.2]. The scalar multiplication over \mathbb{F}_p is carried out according to the double-and-add technique with non-adjacent-form (NAF) representation of the scalar to save some point additions. For ECDSA verification, Shamir’s trick [19] in combination with a joint-sparse-form (JSF) representation of the scalars is used to interleave the two scalar multiplications [19]. We decided to

Table 3. Comparison of MatrixSSL, our SSL, and OpenSSL

Implementation	Number of source files	Lines of code	Size of executable
Original MatrixSSL	30	~ 9,500	114 kB
MatrixSSL with ECC	50	~ 10,900	130–140 kB
OpenSSL 0.9.8	1,100	~ 250,000	2,374 kB

not implement a window method for scalar multiplication because we aimed to keep the memory footprint at a minimum.

Also the algorithms for arithmetic in \mathbb{F}_{2^m} are well documented and rather straightforward to implement. We used the so-called left-to-right comb method with windows of width 4 for the multiplication of binary polynomials [19]. Furthermore, we implemented a generic reduction function for irreducible trinomials and pentanomials. The term generic in this context means that the reduction function accepts arbitrary trinomials and pentanomials as input. In addition, we also included the Montgomery reduction for binary polynomials in our library to support irreducible polynomials which are not trinomials or pentanomials. The scalar multiplication on elliptic curves over \mathbb{F}_{2^m} is performed according to the well-known algorithm of Lopez and Dahab [19].

We actually implemented two versions of the crypto library: one is written entirely in ANSI C, whereas the second contains assembly-language statements to access the custom instructions of our extended LEON-2 core. Reference [13] explains the implementation of the field arithmetic using the CIS instructions in detail. The CIS-optimized version uses Montgomery multiplication for both prime and binary fields. We refrained from the implementation of special reduction techniques for GM primes or sparse irreducible polynomials since we aimed at a “lightweight” implementation of the cryptographic primitives with small code size. The results from [13] indicate that the CIS extensions speed up the multiplication in prime fields by a factor of between two and three, whereas the multiplication in binary fields achieves a six to ten-fold performance gain. The exact speed-up factor depends on a number of implementation options (e.g. loop unrolling) and the length of the operands (e.g. when Karatsuba’s technique [21] is used).

In the following, we evaluate and analyze the handshake performance of the co-designed SSL stack. As mentioned before, our implementation is generic in the sense that it supports arbitrary cipher suites and arbitrary ECC domain parameters. It is, of course, not feasible to evaluate every possible combination of cipher suites and domain parameters in this paper. Therefore, we focus on a representative example, namely an ECC cipher suite that uses ephemeral ECDH for key exchange and ECDSA as signature primitive [5]. We let our co-designed SSL stack operate as server, which means that it has to execute two scalar multiplications to establish a shared secret key. As usual, no client authentication is performed, i.e. the client does not send a certificate to the server.

Figure 1 shows the execution time (in clock cycles) of a scalar multiplication using four NIST prime fields as underlying algebraic structure. All cycle counts were measured on a LEON-2 core with CIS extensions [13], synthesized onto a Xilinx XCV-800 board, clocked at 20 MHz. When using a small field (e.g. a 160 or 192-bit field), the ANSI C version of our crypto library reaches roughly the same performance as OpenSSL, which is a remarkable result when considering that the latter features several performance enhancements such as specialized reduction methods for standardized primes, hand-written assembly code for all performance-critical operations, and code-size increasing optimizations like loop

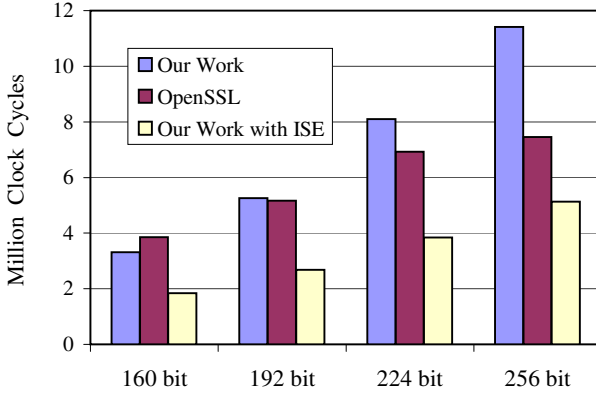


Fig. 1. Performance of scalar multiplication over prime fields

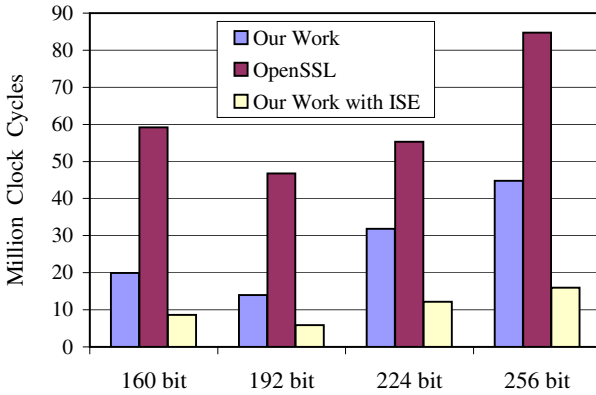


Fig. 2. Performance of entire handshake over prime fields

unrolling). On the other hand, both versions of our library perform the modular reduction according to Montgomery’s algorithm [25], i.e. better timings would be possible when using one of the optimized reduction methods discussed in Section 2.2. The CIS extensions allow one to execute a full scalar multiplication over a 192-bit prime field in $2.6 \cdot 10^6$ cycles. Depending on the field size, the CIS extensions accelerate scalar multiplication by a factor of between 2.0 and 2.5.

Figure 2 illustrates that the performance gained at the field or group level propagates almost lossless all the way up to the application (i.e. the handshake) level. The CIS version of our SSL stack is again by a factor of between 2.0 and 2.5 faster than the ANSI C version that does not use custom instructions for field arithmetic. Our co-design is able to perform a full SSL handshake, from sending the first Hello message until receiving final Finished message, in less than 300 msec on a device running at 20 MHz when using a 192-bit field as underlying algebraic structure. Similar results can also be achieved for binary extension fields of roughly the same order. For comparison, OpenSSL is—depending on

the field type and order—up to a factor of 10 slower than our implementation utilizing the CIS instructions. This big difference is partly due to the efficiency of our field arithmetic and partly due to the lightweight implementation of our protocol stack.

5 Conclusion

We presented a hardware/software co-design of the SSL handshake based on instruction set extensions for the low-level arithmetic operations carried out in public-key cryptography. Our solution offers a significant gain in performance for field arithmetic as well as for an entire handshake when compared with a pure software implementation, thus allowing a handshake over a 192-bit prime field to complete in about 300 msec on a 20 MHz LEON-2 processor equipped with our CIS extensions. A single scalar multiplication over the same field takes approximately $2.6 \cdot 10^6$ cycles when using Montgomery's algorithm for the field arithmetic. Our solution requires very little additional hardware (about 5,500 gates), consumes a negligible amount of additional memory, and allows one to speed up a multitude of cryptographic algorithms, including RSA, DSA, Diffie-Hellman, as well as ECDSA and ECDH over both prime and binary fields. In addition, we have shown that the speed-up achieved in the low-level operations (i.e. the field arithmetic) propagates almost lossless up to the highest layers of the SSL protocol. So, by speeding up field multiplication and squaring using instruction set extensions, the entire high-level SSL handshake can be sped up by almost the same factor.

Acknowledgements. A preliminary version of this paper was presented at the 2nd Workshop on Embedded Systems Security (WESS 2007), which took place in Salzburg, Austria on October 4, 2007.

The research described in this paper has been supported by the EPSRC under grant EP/E001556/1 and, in part, by the European Commission through the ICT Programme under contract ICT-2007-216676 ECRYPT II. The information in this paper reflects only the authors' views, is provided as is, and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Manuel Koschuch, Matthias Hudler and Michael Krüger have been supported by the MA27 – EU-Strategie und Wirtschaftsentwicklung – in the course of the funding programme “Stiftungsprofessuren und Kompetenzteams für die Wiener Fachhochschul-Ausbildungen.”

References

1. Apostolopoulos, G., Peris, V.G., Pradhan, P., Saha, D.: Securing electronic commerce: Reducing the SSL overhead. *IEEE Network* 14(4), 8–16 (2000)
2. Argyroudis, P.G., Verma, R., Tewari, H., O'Mahony, D.E.: Performance analysis of cryptographic protocols on handheld devices. In: *Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications (NCA 2004)*, pp. 169–174. IEEE Computer Society Press, Los Alamitos (2004)

3. Berbecaru, D.G.: On measuring SSL-based secure data transfer with handheld devices. In: Proceedings of 2nd IEEE International Symposium on Wireless Communication Systems (ISWCS 2005), pp. 409–413. IEEE, Los Alamitos (2005)
4. Blake, I.F., Seroussi, G., Smart, N.P.: Advances in Elliptic Curve Cryptography. Cambridge University Press, Cambridge (2005)
5. Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., Möller, B.: Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). Internet Engineering Task Force, Network Working Group, RFC 4492 (May 2006)
6. Brown, M.K., Cheung, D.C., Hankerson, D.R., López Hernández, J.C., Kirkup, M.G., Menezes, A.J.: PGP in constrained wireless devices. In: Proceedings of the 9th USENIX Security Symposium (SECURITY 2000), pp. 247–261. USENIX Association (2000)
7. Coarfa, C., Druschel, P., Wallach, D.S.: Performance analysis of TLS Web servers. ACM Transactions on Computer Systems 24(1), 39–69 (2006)
8. Dierks, T., Rescorla, E.K.: The Transport Layer Security (TLS) Protocol Version 1.1. Internet Engineering Task Force, Network Working Group, RFC 4346 (2006)
9. Freier, A.O., Karlton, P., Kocher, P.C.: The SSL Protocol Version 3.0. Internet Draft (November 1996), <http://wp.netscape.com/eng/ssl3/draft302.txt>
10. Gaisler, J.: The LEON-2 Processor User’s Manual (Version 1.0.10) (January 2003), <http://www.gaisler.com/doc/leon2-1.0.10.pdf>
11. Großschädl, J., Kamendje, G.-A.: Architectural enhancements for Montgomery multiplication on embedded RISC processors. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 418–434. Springer, Heidelberg (2003)
12. Großschädl, J., Savaş, E.: Instruction set extensions for fast arithmetic in finite fields $GF(p)$ and $GF(2^m)$. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 133–147. Springer, Heidelberg (2004)
13. Großschädl, J., Tillich, S., Szekely, A., Wurm, M.: Cryptography instruction set extensions to the SPARC V8 architecture (submitted for publication) (2007)
14. Gupta, V., Gupta, S.: Experiments in wireless internet security. In: Proceedings of the 3rd IEEE Conference on Wireless Communications and Networking (WCNC 2002), vol. 2, pp. 860–864. IEEE, Los Alamitos (2002)
15. Gupta, V., Gupta, S., Chang Shantz, S., Stebila, D.: Performance analysis of elliptic curve cryptography for SSL. In: Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe 2002), pp. 87–94. ACM Press, New York (2002)
16. Gupta, V., Stebila, D., Fung, S., Chang Shantz, S., Gura, N., Eberle, H.: Speeding up secure Web transactions using elliptic curve cryptography. In: Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS 2004), pp. 231–239 (2004)
17. Gura, N., Chang Shantz, S., Eberle, H., Gupta, S., Gupta, V., Finchelstein, D., Goupy, E., Stebila, D.: An end-to-end systems approach to elliptic curve cryptography. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 349–365. Springer, Heidelberg (2003)
18. Gutmann, P.: Performance characteristics of application-level security protocols (2005), http://www.cs.auckland.ac.nz/~pgut001/pubs/app_sec.pdf
19. Hankerson, D.R., Menezes, A.J., Vanstone, S.A.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004)
20. Institute of Electrical and Electronics Engineers (IEEE). IEEE Std 1363-2000: IEEE Standard Specifications for Public-Key Cryptography (August 2000)
21. Karatsuba, A.A., Ofman, Y.P.: Multiplication of multidigit numbers on automata. Soviet Physics - Doklady 7(7), 595–596 (1963)

22. Koç, Ç.K., Acar, T., Kaliski, B.S.: Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro* 16(3), 26–33 (1996)
23. Koschuch, M., Großschädl, J., Payer, U., Hudler, M., Krüger, M.: Workload characterization of a lightweight SSL implementation resistant to side-channel attacks. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) *CANS 2008*. LNCS, vol. 5339, pp. 349–365. Springer, Heidelberg (2008)
24. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton (1996)
25. Montgomery, P.L.: Modular multiplication without trial division. *Mathematics of Computation* 44(170), 519–521 (1985)
26. National Institute of Standards and Technology (NIST). Recommendation for Key Management – Part 1: General (Revised). Special Publication 800-57 (March 2007), <http://csrc.nist.gov/publications/PubsSPs.html>
27. National Security Agency (NSA). NSA Suite B Cryptography. Fact sheet (March 2008), http://www.nsa.gov/ia/programs/suiteb_cryptography/
28. OpenSSL Project. OpenSSL 0.9.7k. (September 2006), <http://www.openssl.org>
29. PeerSec Networks, Inc. MatrixSSL 1.7.1. (2005), <http://www.matrixssl.org>
30. Potlapally, N.R., Ravi, S., Raghunathan, A., Jha, N.K.: A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing* 5(2), 128–143 (2006)
31. Potlapally, N.R., Ravi, S., Raghunathan, A., Lakshminarayana, G.: Optimizing public-key encryption for wireless clients. In: *Proceedings of the 37th IEEE International Conference on Communications (ICC 2002)*, vol. 2, pp. 1050–1056. IEEE, Los Alamitos (2002)
32. Ravi, S., Raghunathan, A., Potlapally, N.R.: Securing wireless data: System architecture challenges. In: *Proceedings of the 15th International Symposium on System Synthesis (ISSS 2002)*, pp. 195–200. ACM Press, New York (2002)
33. Rescorla, E.K.: *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, Reading (2000)
34. Standards for Efficient Cryptography Group (SECG). SEC 1: Elliptic Curve Cryptography (2000), http://www.secg.org/download/aid-385/sec1_final.pdf
35. Wolf, W.H.: Hardware-software co-design of embedded systems. *Proceedings of the IEEE* 28(7), 967–989 (1994)
36. Zhao, L., Iyer, R., Makineni, S., Bhuyan, L.: Anatomy and performance of SSL processing. In: *Proceedings of the 5th International Symposium on Performance Analysis of Systems and Software (ISPASS 2005)*, pp. 197–206. IEEE Computer Society Press, Los Alamitos (2005)

Online/Offline Ring Signature Scheme^{*}

Joseph K. Liu¹, Man Ho Au², Willy Susilo², and Jianying Zhou¹

¹ Cryptography and Security Department
Institute for Infocomm Research, Singapore
{ksliu, jyzhou}@i2r.a-star.edu.sg

² Centre for Computer and Information Security (CCISR)
School of Computer Science and Software Engineering
University of Wollongong, Australia
{aau, wsusilo}@uow.edu.au

Abstract. In this paper, for the first time in the literature, we introduce the notion of online/offline ring signature scheme. Our primitive enables ring signature schemes to be used in practice, since the online mechanism can be performed very efficiently and hence, it is very suitable to be used in a mobile-device environment. We provide a formal model to capture our primitive, and we proceed with a concrete construction of online/offline ring signature schemes. Finally, we show that our scheme is secure in our model.

1 Introduction

Bluetooth is a short-range exchange data enabler protocol that allows mobile devices to communicate in an ad-hoc way. It was originally motivated as the wireless alternative to the RS232 data cable. This technology enables mobile devices, such as iPhone, Windows devices or Android devices, to communicate wirelessly in an ad-hoc manner. It will enable an ad-hoc communication built among business people meeting in a conference room, since the communication can be done efficiently and in a very simple manner. This technology will allow cryptographic techniques to be embedded to it, for instance to create an authenticated message on behalf of the group. One possible solution is by incorporating the primitive put forth by Rivest, Shamir and Tauman known as the ring signature schemes [26]. In fact, this has been “implied” since the invention of ring signatures that these types of primitives can be used on top of ad-hoc technology, such as Bluetooth.

A ring signature scheme (for examples [1, 7, 8, 14, 15, 26, 33, 13, 31, 24, 22, 32, 23, 3, 20, 2, 4, 21, 25]) allows members of a group to sign messages on behalf of the group without revealing their identities, i.e. signer anonymity. In addition, it is not possible to decide whether two signatures have been issued by the same group member. Different from a group signature scheme (for examples, [11], [9] and [5]), the group formation is spontaneous and there is no group manager to

^{*} The first and fourth author of this work are funded by the EU project SMEPP-033563.

revoke the identity of the signer. That is, under the assumption that each user is already associated with a public key of some standard signature scheme, a user can form a group by simply collecting the public keys of all the group members including his own. These diversion group members can be totally unaware of being conscripted into the group.

Ring signature schemes could be used for whistle blowing [26], anonymous membership authentication for ad hoc groups [8] and many other applications which do not want complicated group formation stage but require signer anonymity. For example, in the whistle blowing scenario, a whistleblower gives out a secret as well as a ring signature of the secret to the public. From the signature, the public can be sure that the secret is indeed given out by a group member while cannot figure out who the whistleblower is. At the same time, the whistleblower does not need any collaboration of other users who have been conscripted by him into the group of members associated with the ring signature. Hence the anonymity of the whistleblower is ensured and the public is also certain that the secret is indeed leaked by one of the group members associated with the ring signature.

Ring signature scheme can be used to derive other primitives as well. It had been utilized to construct non-interactive deniable ring authentication [29], perfect concurrent signature [30] and multi-designated verifiers signature [19].

Nevertheless, the existing ring signature schemes requires very heavy computations. Usually the number of exponentiations required during the signing stage is proportional to the number of users included in the ring signature. Say, if the signature includes 10000 users, the signing stage requires at least 10000 exponentiations. This may not be a big problem for personal computers. However, the schemes will not be suitable in practice to be used in mobile devices as these computations will drain the battery quickly.

In this paper, we address the above problem specifically by introducing the notion of “online/offline ring signatures”. In our primitive, the signing stage is divided into two phases. Similar to other online/offline signatures [16, 28, 18, 12, 17, 6], the offline mechanism can be quite computationally heavy, but the online mechanism should be very efficient. This way, we can achieve an ad-hoc communication among mobile devices using the available technology, such as Bluetooth. Our primitive has enabled the use of ring signature schemes in a more practical way.

However, there is one major difference between normal online/offline signatures and online/offline ring signatures. For a normal signature, there is only 1 user. For a ring signature, there are n different users included. The crux of constructing such a scheme relies on the fact that during the offline phase, *the signers are not known* in advance. The group of signers will only be known during the online phase, and therefore this creates some subtleties in the scheme. Otherwise, by using some generic construction of normal online/offline signature schemes such as [28], it is quite trivial to construct one from a normal ring signature scheme.

In the practical point of view, if we need to fix the possible signers in the offline phase, it is not useful. Suppose we need to create a ring signature during a meeting using a mobile device. The offline phase should be done before the meeting. However, it maybe impossible to know who are going to attend the meeting at this moment yet. We are not interested in this model as it is not practical to be used in many applications. Instead, we do not need to know any possible signers in the offline phase. Furthermore, we even allow any third party to generate the offline phase. That is, no secret key is needed and no secret information is generated in the offline phase. This creates even more flexibility for different kinds of scenarios. Therefore it is quite challenging to design and construct such a scheme with these nice properties.

1.1 Contribution

In this paper, we propose a new notion called online/offline ring signature scheme. It is the “online/offline” version of ring signature scheme, with the following additional properties:

1. Most of the heavy computations are done in the offline phase, while the online phase just requires relatively light computation.
2. The online computation requirement is independent to the number of possible signers included in the ring signature. In our construction we just require 2 exponentiations in this phase, no matter how many possible signers are included in the signature.
3. The offline phase does not require the public keys of the possible signers. That is, the public keys are not needed to be fixed at this phase yet.
4. The offline phase can be done by any third party. All information produced or generated during this phase is publicly known. There is no secret information included or required in this phase.

Our scheme is proven secure in the random oracle model, under the standard RSA assumption.

2 Definitions

2.1 Mathematical Assumption

The security of our scheme relies on the RSA assumption with safe prime, which is defined as follow:

Definition 1 (Safe Prime). *p is a safe prime if it can be expressed as $2p' + 1$ where p' is also a prime.*

Definition 2 (RSA Assumption with Safe Prime). *Let $N = pq$ where p and q are k -bits length safe primes. Let e be a number such that e and $\phi(N)$ are co-prime. Given an element $r \in \mathbb{Z}_N$ chosen at random, find an integer x such that $x^e = r \pmod N$. An adversary \mathcal{A} has at least an ϵ advantage if*

$$\Pr[\mathcal{A}(N, e, r) = x \mid x^e = r \pmod N] \geq \epsilon$$

We say that the (ϵ, τ, k) -RSA assumption holds if no algorithm running in time at most τ can solve that RSA problem with advantage at least ϵ , where the modulus is a product of two safe primes with k -bits length.

2.2 Security Definition

Definition 3. A online/offline ring signature scheme is defined by the following algorithms:

- **Key-Gen** is a probabilistic algorithm taking as input a security parameter. It returns the user secret key sk and public key pk .
- **Offline-Sign** is a probabilistic algorithm taking n' as input, where n' is the maximum number of users to be included in the ring signature. Optionally, it may also take the actual signer's secret key sk and public key pk as input. It returns an offline signature $\bar{\sigma}$.
- **On-Sign** is a probabilistic algorithm taking $(L, m, sk, \bar{\sigma})$ as input, where L is the list of n public keys to be included in the ring signature and $n \leq n'$ and m is the message to be signed. It returns a signature σ .
- **Verify** is a deterministic algorithm taking (L, m, σ) as input. It outputs either **Accept** or **Reject**.

The security of a ring signature scheme consists of two requirements, namely *Signer Ambiguity* and *Existential Unforgeability*. They are defined as follows.

Definition 4 (Signer Ambiguity). Let $L = \{pk_1, \dots, pk_n\}$ be the list of public keys and $L_{sk} = \{sk_1, \dots, sk_n\}$ be the corresponding secret keys. Each key is generated by **Key-Gen**. A ring signature scheme is said to be unconditionally signer ambiguous if, for any L , any message m , and any signature $\sigma \leftarrow \text{On-Sign}(L, m, sk_\pi, \text{Offline-Sign}(|L|))$ where $sk_\pi \in L_{sk}$, any unbound adversary \mathcal{A} accepts as inputs L , m and σ , outputs π with probability $1/n$.

It means that even all the private keys are known, it remains uncertain that which signer out of n possible signers actually generates a ring signature.

Definition 5 (Existential Unforgeability). For a ring signature scheme with n public keys, the existential unforgeability is defined as the following game between a challenger and an adversary \mathcal{A} :

1. The challenger runs algorithm **Key-Gen**. Let $L = \{pk_1, \dots, pk_n\}$ be the set of n public keys and $L_{sk} = \{sk_1, \dots, sk_n\}$ be the corresponding secret keys. \mathcal{A} is given L .
2. \mathcal{A} can adaptively queries the signing oracle q_S times: On input any message m and L' where $L' \subseteq L$ (the corresponding secret keys are denoted by L'_{sk}) returns a ring signature $\sigma \leftarrow \text{On-Sign}(L', m, sk_\pi, \text{Offline-Sign}(|L'|))$, where $sk_\pi \in L'_{sk}$ and $\text{Verify}(L', m, \sigma) = \text{Accept}$.
3. Finally \mathcal{A} outputs a tuple (L^*, m^*, σ^*) .

\mathcal{A} wins the game if:

1. $L^* \subseteq L$.
2. (L^*, m^*) has not been submitted to the signing oracle.
3. $\text{Verify}(L^*, m^*, \sigma^*) = \text{Accept}$

We define \mathcal{A} 's advantage in this game to be $\text{Adv}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

3 The Proposed Scheme

3.1 Construction

Let k_1, k_2, k_3 be security parameters such that $k_1 \leq k_2 - 1$. Assume G is a hash function that maps any arbitrary string into k_1 -bits odd integer.

Key-Gen: Each user selects two safe primes p, q of length k_2 -bits, such that $p = 2p' + 1, q = 2q' + 1$ where p', q' are also primes. His secret key is (p, q) and public key is $N = pq$.

Offline-Sign: Assume there are maximum n' users to be included in the ring signature (their public keys are not yet known in this stage, except the real signer). For $i = 1, \dots, n' - 1$, randomly selects integers $x_i \in_R \{0, 1\}^{2k_2 - 1}$, $e_i \in_R \{0, 1\}^{k_3}$ and computes $y_i = x_i^{G(e_i)}$ (without modulus). Stores the offline signature $\bar{\sigma} = (x_1, e_1, y_1, \dots, x_{n'-1}, e_{n'-1}, y_{n'-1})$.

Online-Sign: Let $L = \{N_1, \dots, N_n\}$ be a list of n public keys to be included in the ring signature, where $n \leq n'$. Let $H_i : \{0, 1\}^* \rightarrow \mathbb{Z}_{N_i}$ be some hash functions for $i = 1, \dots, n$. W.l.o.g., we assume user n is the actual signer. The actual signer executes the following steps:

1. Randomly generates an integer $e_n \in_R \{0, 1\}^{k_3}$ and computes $d_n = 1/G(e_n) \bmod \phi(N_n)$.
2. Randomly generates $r_n \in_R \mathbb{Z}_{N_n}$, computes $c_1 = H_1(L, m, r_n)$.
3. For $i = 1, \dots, n - 1$, computes $c_{i+1} = H_{i+1}(L, m, c_i + y_i \bmod N_i)$.
4. Computes $x_n = (r_n - c_n)^{d_n} \bmod N_n$.

If $|x_n| = 2k_2$ bits, repeats step 2 - 4 until getting another x_n which is strictly less than $2k_2$ bits. Outputs the signature $\sigma = (x_1, e_1, \dots, x_n, e_n, c_1)$.

Verify: To verify a signature for message m and public keys $L = \{N_1, \dots, N_n\}$, For $i = 1, \dots, n$ computes $r_i = c_i + x_i^{G(e_i)} \bmod N_i$ and $c_{i+1} = H_{i+1}(L, m, r_i)$ if $i \neq n$. Accept if $c_1 = H_1(L, m, r_n)$. Otherwise reject.

Remarks

1. The offline signing phase can be executed by any third party. We do not require the secret key of the actual signer. There is neither any secret information (such as secret randomness) produced at this stage. All data generated here are publicly known. The tradeoff is, we require a modulus inverse

$1/G(e_n)$ in the online part. To further improve efficiency, this part (step 1 of the online signing stage) can be put into the offline signing phase. However, since this step requires the knowledge of the secret key (factorization of N_n), by doing so, the offline signing phase also requires the secret key as the input and (e_n, d_n) are stored as part of the offline signature.

2. The expected running time of step 2 - 4 in Online-Sign is 2. It is calculated as follow: For each time,

$$\Pr[|x_n| = 2k_2] \leq 0.5$$

as $|N_n| = 2k_2$. That means at least with probability 0.5 the computation of step 4 is successful ($|x_n| < 2k_2$). The expected running time S should be:

$$S = 1 \times \frac{1}{2} + 2 \times \frac{1}{2^2} + 3 \times \frac{1}{2^3} + 4 \times \frac{1}{2^4} + \dots \tag{1}$$

From (1), multiply both sides by $\frac{1}{2}$, we get

$$\frac{1}{2}S = \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \frac{4}{2^5} + \dots \tag{2}$$

(1) - (2), we get

$$\frac{1}{2}S = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \dots = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = 1 \tag{3}$$

From (3), we can get the result $S = 2$.

3.2 Security Analysis

Theorem 1. *Our ring signature scheme is unconditionally signer ambiguous.*

Proof. All e_i are taken randomly from $\{0, 1\}^{k_3}$ and all x_i except x_n are also taken randomly from $\{0, 1\}^{2k_2-1}$. At the closing point, $x_n \in \{0, 1\}^{2k_2-1}$ also distributes randomly as r_n is randomly chosen, c_n depends on previous x_i and e_i which are all random numbers. The remaining c_1 is uniquely determined from L, m and r_n .

Note that the reason why we need to restrict x_n to be at most $2k_2 - 1$ bits is that, as other x_i are all at most $2k_2 - 1$ bits, if x_n is $2k_2$ bits, one may know that user n is the actual signer. □

Theorem 2. *Suppose the (τ', ϵ', k_2) -RSA assumption with safe prime holds. Then our ring signature scheme with n users is $(\tau, q_s, q_h, q_g, \epsilon)$ -secure against existential forgery under an adaptive chosen message attack provided that:*

$$\epsilon' \geq \frac{2 \left(1 - \frac{q_h q_s}{N_{min}}\right) \left(1 - \frac{1}{N_{min}}\right) \left(1 - \frac{1}{2^{k_2-1}}\right) \epsilon}{q_h(q_h + 1)q_g n} \quad \tau' = \tau$$

where N_{min} is the smallest modulus among n public keys, q_s is the maximum number of signing oracle queries allowed, q_h is the maximum number of H_i random oracle queries allowed, q_g is the maximum number of G random oracle queries allowed.

Proof. Setup: The proof uses the approach described in [11]. Suppose the adversary \mathcal{A} can forge the ring signature scheme with n users. We construct an algorithm \mathcal{S} that uses \mathcal{A} to solve the (τ', ϵ', k_2) -RSA problem.

\mathcal{S} receives the problem instance (N, e, r) , where N is the product of two safe prime numbers of length k_2 -bits, e is coprime to $\phi(N)$, $2 < e < N$ and $0 \leq r < N$. \mathcal{S} is asked to output an integer x such that $x^e = r \pmod N$. For simplicity, we let $k_1 = k_2 - 1$ to be the length of the output string of G random oracle.

\mathcal{S} randomly chooses $\pi \in_R [1, n]$ and assigns the public key of user π to be N (the problem instance). For the other $n - 1$ users' public keys, \mathcal{S} generates them according to the algorithm.

\mathcal{S} also chooses three integers u, v, w such that $1 \leq u \leq v \leq q_h$ and $1 \leq w \leq q_g$.

Oracle Simulation:

- *H_i Random Oracle:* For simplicity, the H_i random oracles are treated as single oracle that takes $Q_j = (i, L_j, m_j, r_j)$ as the j -th query and returns a random value that corresponds to $H_i(L_j, m_j, r_j)$ maintaining consistency against duplicated queries.
- *G Random Oracle:* \mathcal{S} assigns e (the problem instance) to be the output of the w -th query. For the other queries, it just returns a random value and maintaining consistency against duplicated queries.
- *Signing Oracle:* Upon receiving the signing query for (L_j, m_j) , \mathcal{S} simulates the signing oracle in the following way.
 1. Randomly choose $c_1 \in_R \mathbb{Z}_{N_1}$.
 2. For $i = 1, \dots, |L_j|$, randomly select integers $x_i \in_R \{0, 1\}^{2k_2-1}$ and $e_i \in_R \{0, 1\}^{k_3}$, compute $r_i = x_i^{G(e_i)} + c_i \pmod{N_i}$, and then compute $c_{i+1} = H_{i+1}(L_j, m_j, r_j)$ if $i \neq |L_j|$.
 3. Assign c_1 to the value of $H_1(L_j, m_j, r_{|L_j|})$.

Output Calculation: Since the queries form a ring, there exists at least one index, say κ , in $\{1, \dots, n\}$ such that $Q_u = (\kappa + 1, L, m, r_\kappa)$ and $Q_v = (\kappa, L, m, r_{\kappa-1})$ satisfy $u \leq v$. Namely, κ is in between the gap of query order. We call such (u, v) a gap index. Note that $u = v$ happens only if $n = 1$, which means that the resulting L contains only one public-key. If there are two or more gap indices with regard to a signature, only the smallest one is considered.

At the beginning of the simulation, \mathcal{S} has chosen a pair of index (u, v) randomly such that $1 \leq u \leq v \leq q_h$. If the guess is correct, \mathcal{S} receives $Q_u = (\kappa + 1, L, m, r_\kappa)$ and $Q_v = (\kappa, L, m, r_{\kappa-1})$ so that (u, v) is a gap index. When query Q_v is made (u -th query has been already made by this moment), \mathcal{S} returns $c_\kappa = r_\kappa - r \pmod{N_\kappa}$ (r is the problem instance) as the value of $H_\kappa(L, m, r_{\kappa-1})$. If \mathcal{A} is successful in forgery, it outputs x_κ that satisfies $r_\kappa = c_\kappa + x_\kappa^{G(e_\kappa)} \pmod{N_\kappa}$. Since $r_\kappa = c_\kappa + r \pmod{N_\kappa}$, we obtain x_κ as the inverse of r with regard to N , if e_κ is the w -th G -query. That is, the output of that particular query is e (the problem instance).

Probability Analysis: \mathcal{S} is successful if

1. \mathcal{A} outputs a valid forged signature;
2. There is no abortion or failure in any oracle simulation; and
3. All guesses are correct.

\mathcal{A} outputs a valid forged signature with probability ϵ .

\mathcal{S} fails if Step 3 in the signing oracle simulation causes inconsistency in H_1 . It happens with probability at most q_h/N_{min} where N_{min} is the smallest N_i in L . Hence, the simulation is successful q_s times with probability at least

$$\left(1 - \frac{q_h}{N_{min}}\right)^{q_s} \geq 1 - \frac{q_h q_s}{N_{min}}$$

For H_i random oracle, with probability at least $1 - 1/N_{min}$, there exist queries $Q_j = (i + 1, L, m, r_i)$ for all $i = 1, \dots, n$ due to the ideal randomness of H . Similarly, for G random oracle, with probability at least $1 - 1/2^{k_2-1}$, there will be no collision occur.

At the beginning of the simulation, \mathcal{B} selects a pair of index (u, v) . With probability $2/q_h(q_h + 1)$, the guess is correct. \mathcal{B} also selects an index w for the G random oracle query, whose output is assigned to the problem instance. With probability $1/q_g$, the guess is correct. \mathcal{B} needs to guess the index of the user corresponding to the (u, v) gap. \mathcal{B} is correct if $\pi = \kappa$. This happens with probability $1/n$.

Combining all cases, overall successful probability of \mathcal{B} is at least

$$\frac{2\left(1 - \frac{q_h q_s}{N_{min}}\right)\left(1 - \frac{1}{N_{min}}\right)\left(1 - \frac{1}{2^{k_2-1}}\right)\epsilon}{q_h(q_h + 1)q_g n}$$

The running time of \mathcal{S} is almost the same as τ as \mathcal{S} runs \mathcal{A} only once and the simulation cost for the signing oracle and the random oracles are assumed to be sufficiently smaller than τ . \square

4 Efficiency of Existing Ring Signatures

The following table (Table 1) summarizes the time complexities of existing ring signatures. We breakdown the time complexity of the protocol into the number of multi-exponentiations (multi-EXPs). A multi-EXP computes the product

Table 1. Time Complexities of Existing Ring Signatures

Scheme	Number of Multi-EXP
Rivest-Shamir-Tauman [26]	$n + 1$
Abe-Ohkubo-Suzuki [1]	$n + 1$
Dodis-Kiayias-Nicolosi-Shoup [15]	14
Chow-Wei-Liu-Yuen [13]	n
Shacham-Waters [27]	$2n + 2$
Chandran-Groth-Sahai [10]	$5 + 6\sqrt{n} + \frac{n+1}{3}$
Our scheme	2

of exponentiations faster than performing the exponentiations separately. Normally, a multi-based exponentiation takes only 10% more time compared with a single-based exponentiation. We assume that one multi-EXP operation multiplies up to 3 exponentiations. Let n be the size of the ring and the number of multi-EXP is taken after the n public keys and the message are known.

5 Conclusion

In this paper, we have proposed a new notion called Online/Offline Ring Signature scheme. Under this notion, most of the heavy computations are done in the offline phase. At this phase the public keys of all users and the message to be signed are yet to be known. In the online phase, only very little computations are needed after knowing those public keys and the signing message. We provided a concrete construction of this notion. In our construction, the offline phase can be done by other third party. This allows more flexibility. We believe the online/offline ring signature scheme can be used in many different applications such as authentication or whistle blowing using mobile devices.

There are some future improvements that can be done. One of them is to further reduce the online computation requirement. Currently we still require about 2 exponentiations during the online phase. It is better to eliminate it totally. Another open problem is to construct a constant size online/offline ring signature scheme as the signature size of our current construction is still linear with the number of users included in the ring. Finally, it would be interesting to construct an online/offline ring signatures that do not require random oracle models.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of- n Signatures from a Variety of Keys. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 415–432. Springer, Heidelberg (2002)
2. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Constant-size ID-based linkable and revocable-iff-linked ring signature. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 364–378. Springer, Heidelberg (2006)
3. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Certificate based (linkable) ring signature. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 79–92. Springer, Heidelberg (2007)
4. Au, M.H., Liu, J.K., Yuen, T.H., Wong, D.S.: ID-based ring signature scheme secure in the standard model. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 1–16. Springer, Heidelberg (2006)
5. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
6. Boneh, D., Boyen, X.: Short signatures without random oracles the SDH assumption in bilinear groups. *Journal of Cryptology* 2, 149–177 (2008)

7. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
8. Bresson, E., Stern, J., Szydło, M.: Threshold Ring Signatures and Applications to Ad-hoc Groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002)
9. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups (Extended Abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
10. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007)
11. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
12. Chen, X., Zhang, F., Susilo, W., Mu, Y.: Efficient generic online/offline signatures without key exposure. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 18–30. Springer, Heidelberg (2007)
13. Chow, S.S., Liu, J.K., Wei, V.K., Yuen, T.H.: Ring Signatures without Random Oracles. In: ASIACCS 2006, pp. 297–302. ACM Press, New York (2006)
14. Chow, S.S.M., Yiu, S.-M., Hui, L.C.K.: Efficient Identity Based Ring Signature. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 499–512. Springer, Heidelberg (2005); Also available at Cryptology ePrint Archive, Report 2004/327
15. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous Identification in Ad Hoc Groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
16. Even, S., Goldreich, O., Micali, S.: On-line/Off-line digital signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 263–275. Springer, Heidelberg (1990)
17. Joye, M.: An efficient on-line/off-line signature scheme without random oracles. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 98–107. Springer, Heidelberg (2008)
18. Kurosawa, K., Schmidt-Samoa, K.: New online/offline signature schemes without random oracles. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 330–346. Springer, Heidelberg (2006)
19. Laguillaumie, F., Vergnaud, D.: Multi-designated Verifiers Signatures. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 495–507. Springer, Heidelberg (2004)
20. Liu, D.Y.W., Liu, J.K., Mu, Y., Susilo, W., Wong, D.S.: Revocable ring signature. *J. Comput. Sci. Technol.* 22(6), 785–794 (2007)
21. Liu, J.K., Susilo, W., Wong, D.S.: Ring signature with designated linkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 104–119. Springer, Heidelberg (2006)
22. Liu, J.K., Wei, V.K., Wong, D.S.: A Separable Threshold Ring Signature Scheme. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 352–369. Springer, Heidelberg (2004)
23. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract). In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004)

24. Liu, J.K., Wong, D.S.: On the Security Models of (Threshold) Ring Signature Schemes. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 204–217. Springer, Heidelberg (2005)
25. Liu, J.K., Wong, D.S.: Enhanced security models and a generic construction approach for linkable ring signature. *Int. J. Found. Comput. Sci.* 17(6), 1403–1422 (2006)
26. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
27. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 166–180. Springer, Heidelberg (2007)
28. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
29. Susilo, W., Mu, Y.: Non-Interactive Deniable Ring Authentication. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 386–401. Springer, Heidelberg (2004)
30. Susilo, W., Mu, Y., Zhang, F.: Perfect Concurrent Signature Schemes. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 14–26. Springer, Heidelberg (2004)
31. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable Linkable Threshold Ring Signatures. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 384–398. Springer, Heidelberg (2004)
32. Wong, D.S., Fung, K., Liu, J.K., Wei, V.K.: On the RS-Code Construction of Ring Signature Schemes and a Threshold Setting of RST. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 34–46. Springer, Heidelberg (2003)
33. Zhang, F., Kim, K.: ID-Based Blind Signature and Ring Signature from Pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002)

Policy-Controlled Signatures^{*}

Pairat Thorncharoensri, Willy Susilo, and Yi Mu

Centre for Computer and Information Security
School of Computer Science & Software Engineering
University of Wollongong, Australia
{pt78,wsusilo,ymu}@uow.edu.au

Abstract. In this paper, we present a new cryptographic primitive called “policy-controlled signature”. In this primitive, a signer can sign a message and attach some policies to it. Only a verifier who satisfies the policies attached can verify the authenticity of the message. This type of signature schemes has many applications, in particular to deal with sensitive data, where the signer does not want to allow anyone who is not authorized to verify its authenticity. Nonetheless, there is no existing cryptographic primitives that can offer this feature in the literature. Policy-controlled signatures can be seen to be similar to the notion of designated verifier signatures, as it can also be used to *designate* a signature to multiple recipients. When there is only a single attribute involved in a policy presented by a verifier, then we will achieve a designated verifier signature (with some trivial modifications). Therefore, policy-controlled signatures can be viewed as the generalization of the notion of the designated verifier signatures. We present a formal model to capture this notion. Furthermore, we also present a concrete scheme that is secure in our model. Finally, we briefly mention about an implementation that incorporates P3P to realize policy-controlled signatures.

1 Introduction

Consider the following scenario. Alice is a CIA agent. She would like to convey a sensitive message in regards to the international terrorism to the other secret agents, but this message should not be verifiable by the public. Therefore, this message should be verifiable by $\{\text{CIA agents} \vee \text{KGB agents} \vee (\text{secret agents in the country} \wedge \text{authorized agents by the US Government})\}$. The first two conditions imply that if the agent is either a CIA agent or a KGB agent, then the message should be verifiable. The third condition implies that if the person is a secret agent in the country where he/she is authorized by the US government, then he/she should be able to verify the message as well. The message should not be verifiable by any other people outside this authorized set. Furthermore, it is also required that upon the verification of the message by the authorized agents, the agents cannot *relay* this conviction and convince any other third party outside the authorized set. This is a typical scenario where policy-controlled signatures

^{*} This work is partially supported by ARC Linkage Project Grant LP0667899.

are useful. Furthermore, there are many other applications that involve contents sensitive, such as medical records, that are only authorized to several people such as medical doctors, etc.

The above scenario seems to be straightforward and it could be solved by using the existing cryptographic primitives. Nonetheless, we shall demonstrate that unfortunately, the existing cryptographic primitives cannot be used to solve this problem. Firstly, the notion that is *close* to this scenario is the notion of policy-based cryptography [1]. In a policy-based cryptography, the sender is equipped with the policy, but not the verifier. Unlike our scenario, policy-based cryptography introduced in [1] allows the sender to sign a message correctly if and only if the sender satisfies some policies. We argue that our scenario is more natural compared to the scenario used in policy-based cryptography [1] since usually the sender should specify which receiver(s) that should be able to verify the authenticity of the message. At a glance, we may be able to achieve the above solution by signing a message with a regular signature scheme, and then encrypt it with a policy-based encryption scheme. Nevertheless, in this solution, a verifier who can decrypt the ciphertext can obtain the signature and make it publicly verifiable. Hence, it violates the requirement as stated above.

At the first glance, this notion seems to be closely related to designated verifier signatures [2]. With a trivial modification, policy-controlled signatures are in fact the generalization of the notion of designated verifier signatures. If the number of verifiers is merely only *one*, then policy-controlled signatures with a trivial modification will achieve a designated verifier signature scheme as defined in [2]. Nevertheless, policy-controlled signatures allow multiple verifiers to verify the signature on some message (and hence, it provides a *similar* property as the designated *multiple* verifiers signatures). The idea is to allow several receivers to satisfy some policies, and therefore, the signature produced by the signer should be verifiable by these receivers. Nevertheless, the other party who does not satisfy the policies should not be able to verify the signature.

This notion resembles a similar idea to the notion of multi-designated verifier signatures. However, in the latter, the collaborations of the verifiers are required in verifying the designated signatures. In contrast to this notion, in policy-controlled signatures, each verifier can verify it individually, as long as it holds the satisfying policies.

Due to the requirements of the verifier to satisfy some policies specified by the signer, we call our primitive as *policy-controlled signatures*. We further argue that this cryptographic primitive is hard to construct. In particular, we should ensure that any coalition of unauthorized verifiers must not be able to verify the authenticity of any signature. If coalition resistance is not required, then we can simply assign a separate policy for each verifier to enable the verifier to test the authenticity of the signature.

This notion can be viewed as the *dual* of secret handshakes, as introduced in [3]. Secret handshakes aim to allow members of the group to identify each other. Secret handshakes ensure that non-group members cannot recognize the handshake and hence are not able to recognize group members. Additionally,

non-group members cannot perform the handshake and hence are unable to trifle group members into thinking they are also members [3]. Secret handshakes are merely encryption schemes that only allow the group members to decrypt the ciphertext.

1.1 Related Work

Since the seminal introduction of digital signature notion [4] and its formalization [5], the notion of digital signatures have been extended to capture different scenarios and situations in real life. Among the different type of signatures, we include some variants of digital signatures that are related to our notion, that include (universal) designated verifier signature [6,7,8,9,10,11,12,13] and policy-based cryptography [1,14,15,16].

The notion of designated verifier signatures was put forth by Jakobsson, Sako and Impagliazzo in [2]. In this notion, a signature ensures authenticity and deniability properties at the same time. The deniability property stops further conviction of a validity of a digital signature by a verifier to any other third party. The authenticity is ensured by the signature since the verifier can be sure that the signature is indeed created by the original signer, but nobody else will be convinced with this fact. Recently, the topics on designated verifier signatures have been actively studied [6,7,8,9,10,11].

The notion of ring signatures was introduced and formalized by Rivest, Shamir and Tauman [17] with the aim to provide an unconditional security to the signer in a group. In this notion, a signer (alone without any cooperation with other signers) can generate a signature that is verifiable to have been signed by one of the signers in a ring (a set of signers). Hence, from the verifier's point of view, a ring signature provides authentication of a message by one signer in the ring. Two-party ring signatures are known to give the construction of designated verifier signatures. This notion has also been actively studied in the literature [18,19,20,21,22,23,24,25,26]. In particular, the notion of threshold ring signatures, which provides the integrity of the message, and the authenticity, non-repudiation and anonymity of the multi-signers, was first formalized by Bresson, Stern and Szydlo in [18]. Similar to ring signature schemes, threshold ring signature schemes support multi-signers (cf. the original ring signature schemes).

The notion of policy-based cryptography, which includes policy-based encryption schemes and policy-based signature schemes, was firstly introduced by Bagga and Molva in [1]. In this notion, a sender (or signer, resp.) can only construct an encrypted message (or sign a message, resp.) if he/she satisfies the required policy. A policy-based encryption scheme provides the authorization and the message's integrity, while a policy-based signature provides the message's integrity, and the authenticity and non-repudiation of signer. Bagga and Molva gave an improved construction of policy-based encryption in [14]. This improved policy-based encryption is enhanced from the previous construction by including a public key of user into a user's credential. Here, the user is required to have a credential and a public key to decrypt the ciphertext. The purpose of this improvement is to prevent the collision attack.

1.2 Our Contributions

In this paper, we introduce the notion of policy-controlled signature (PCS) schemes. Our notion allows a receiver to verify the authenticity of the signed message if and only if the receiver satisfies the policy specified by the sender (or signer). We formalize this notion and define its security model and requirements. Furthermore, we instantiate our model with a concrete construction that is proven secure in our model.

Paper Organization

The paper is organized as follows. In the next Section, we will review some preliminaries that will be used throughout this paper. The definition of PCS and its security notions will be presented in Section 3. Next, our concrete scheme will be provided in Section 4.2. Then, proof of the security of our concrete scheme is presented in Section 5. In Section 6, we briefly demonstrate our implementation that incorporates P3P [27]. Finally, we conclude the paper.

2 Preliminaries

2.1 Notation

We will use the following notations throughout this paper. Let PPT denote a probabilistic polynomial-time algorithm. When a PPT algorithm F privately accesses and executes another PPT algorithm E , we denote it by $F^{E(\cdot)}(\cdot)$. We denote by $poly(\cdot)$ a deterministic polynomial function. For all polynomials $poly(k)$ and for all sufficiently large k , if $q \leq poly(1^k)$ then we say that q is polynomial-time in k . We say that a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* if, for all constant $c > 0$ and for all sufficiently large n , $f(n) < \frac{1}{n^c}$. Denote by $l \stackrel{\$}{\leftarrow} L$ the operation of picking l at random from a (finite) set L . A collision of a function $h(\cdot)$ refers to the case when there are message pair m, n of distinct points in its message space such that $h(m) = h(n)$. We denote by \parallel the concatenation of two strings (or integers).

2.2 Bilinear Pairing

Let \mathbb{G}_1 and \mathbb{G}_2 be cyclic multiplicative groups generated by g_1 and g_2 , resp. The order of both generators is a prime p . Let \mathbb{G}_T be a cyclic multiplicative group with the same order p . Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear mapping with the following properties:

1. *Bilinearity*: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$.
2. *Non-degeneracy*: There exists $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $\hat{e}(g_1, g_2) \neq 1$.
3. *Computability*: There exists an efficient algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$.

Note that there exists $\varphi(\cdot)$ function which maps \mathbb{G}_1 to \mathbb{G}_2 or vice versa in one time unit.

2.3 Complexity Assumptions

Definition 1 (Computation Diffie-Hellman (CDH) Problem). *Given a 3-tuple $(g, g^x, g^y \in \mathbb{G}_1)$ as input, output $g^{x \cdot y}$. An algorithm \mathcal{A} has advantage ϵ' in solving the CDH problem if*

$$\Pr [\mathcal{A}(g, g^x, g^y) = g^{x \cdot y}] \geq \epsilon'$$

where the probability is over the random choice of $x, y \in \mathbb{Z}_q^*$ and the random bits consumed by \mathcal{A} .

Assumption 1 ((t, ϵ') -Computation Diffie-Hellman Assumption). We say that the (t, ϵ') -CDH assumption holds if no PPT algorithm with time complexity $t(\cdot)$ has advantage at least ϵ' in solving the CDH problem.

Definition 2 (Decision Bilinear Diffie-Hellman (DBDH) Problem)

Given a random 4-tuple $(g, g^a, g^b, g^c) \in \mathbb{G}_1$ and a random integer $Z \in \mathbb{G}_T$ as input, decide whether or not $Z = \hat{e}(g, g)^{abc}$. An algorithm \mathcal{A} is said to (τ, ϵ') solves the DBDH problem in $\mathbb{G}_1, \mathbb{G}_T$, if \mathcal{A} runs in time τ , and

$$\begin{aligned} & |\Pr [\mathcal{A}(g, g^a, g^b, g^c, Z = \hat{e}(g, g)^{abc}) = 1] - \Pr [\mathcal{A}(g, g^a, g^b, g^c, Z = \hat{e}(g, g)^d) = 1]| \\ & \geq \epsilon', \end{aligned}$$

where the probability is taken over the random choices of $a, b, c, d \in \mathbb{Z}_p$, $g \in \mathbb{G}_1$, and the random bits consumed by \mathcal{A} .

Assumption 2 (Decision Bilinear Diffie-Hellman Assumption). We say that the (t, ϵ') -DBDH assumption in $\mathbb{G}_1, \mathbb{G}_T$ holds if there is no PPT algorithm that (t, ϵ') solves the DBDH problem.

3 Policy-Controlled Signature Schemes (PCS)

Model

Let TA denote a trusted authority who issues credentials associated with policies. Let CA denote a certificate authority who generates system parameters and certifies public keys for all parties. There are two main players in a policy-controlled signature scheme, namely a signer and a verifier. A signer S generates a signature that can be verified *only* when a verifier V holds a credential satisfying the policy. V holds credentials issued by TA .

Let A denote an assertion issued by TA . Each assertion A may be a hash value of some statements, such as “CIA agent”. We define POL to be a policy which contains a set of assertions $POL = \bigwedge_{i=1}^a [\bigvee_{j=1}^{a_i} [\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ where i, j, k are indexes. In general, a policy POL can be represented in the disjunctive normal form (DNF) or the conjunctive normal form (CNF) or any combination of both forms. The policy POL is in the DNF when $a = 1$ and in CNF when $\forall i, \forall j : a_{i,j} = 1$. For example, a policy in DNF is as follows $POL = “(A_{1,1,1} \wedge A_{1,1,2}) \vee (A_{1,2,1} \wedge A_{1,2,2}) \vee A_{1,3,1} \vee (A_{1,4,1} \wedge A_{1,4,2})”$.

For simplicity, let $VCR_{i,j,k}$ be a credential for an assertion and let $\overline{POL} = [\{VCR_{1,1,1}, \dots, VCR_{a,1,a_i,1}\}, \dots, \{VCR_{1,a_i,1}, \dots, VCR_{a,a_i,a_i,j}\}]$ be a set of the entire possible set of credential that satisfy the policy POL , where i, j, k are indexes for assertion associated to credential. Let $\{VCR_{i,j,k}\} = VCR_{1,j,1}, \dots, VCR_{a,j,a_i,j}$ be a set of credentials, which it may or may not be a set of credentials in \overline{POL} . Let $\{VCR\} = VCR_{1,1,1}, \dots, VCR_{a,a_i,a_i,j}$ be the entire credentials, where i, j are indexes.

Without losing generality, we assume that all parties must comply with the registration protocol with a certificate authority CA to obtain a certificate on their respective public keys. In the following, we provide a definition of policy-controlled signature scheme as follows.

Definition 3. A policy-controlled signature scheme Σ is an 6-tuple (Setup, TKeyGen, SKeyGen, CreGen, Sign, Verify) such that

Signature Scheme Setup:

- System Parameters Generation (Setup):
Setup is a PPT algorithm that, on input a security parameter \mathcal{K} , outputs the system parameters param .
- TA Key Generator (TKeyGen) :
TKeyGen is a PPT algorithm that, on input the system parameters param , outputs strings (sk_{TA}, pk_{TA}) where they denote a secret key and a public key of trusted authority, respectively. That is $\{pk_{TA}, sk_{TA}\} \leftarrow TKeyGen(\mathit{param})$.
- Signer Key Generator (SKeyGen) :
SKeyGen is a PPT algorithm that, on input the system parameters param and a public key of the trusted authority pk_{TA} , outputs strings (sk_S, pk_S) where they denote a secret key and a public key of a signer, respectively. That is $\{pk_S, sk_S\} \leftarrow SKeyGen(\mathit{param}, pk_{TA})$.
- Verifier Credential Generator (CreGen) :
CreGen is a PPT algorithm that, on input the system parameters param , the TA's public key, the policy POL , outputs verifier credential strings $\{VCR_{i,j,k}\}$ where i, j, k are an index of credential strings. That is $\{VCR_{i,j,k}\} \leftarrow CreGen(\mathit{param}, pk_{TA}, POL)$.

Policy-controlled Signature Signing (Sign):

On input the system parameters param , the trust authority's public key pk_{TA} , the signer's secret key sk_S , the signer's public key pk_S , a message M and the policy POL , Sign outputs signer's signature σ . That is $\sigma \leftarrow Sign(\mathit{param}, M, sk_S, pk_S, pk_{TA}, POL)$.

Policy-controlled Signature Verification (Verify):

On input the system parameters param , the trust authority's public key pk_{TA} , the signer's public key pk_S , the policy POL , a set of credentials $\{VCR_{i,j,k}\} \in \overline{POL}$, a message M and a signature σ , Verify outputs a verification decision $d \in \{\text{Accept}, \text{Reject}\}$. That is $d \leftarrow Verify(\mathit{param}, M, \sigma, pk_{TA}, pk_S, POL, \{VCR_{i,j,k}\})$.

3.1 Security Model

Queries: To model the ability of the adversaries in breaking the security of PCS schemes, the following queries is prescribed.

\mathcal{SSO} oracle: At most q_{SS} , \mathcal{A} can make a query for a signature σ on its choice of a message M . As a response, \mathcal{SSO} runs the *Sign* algorithm to generate a signature σ on a message M corresponding with pk_{TA} , pk_S and POL . \mathcal{SSO} then returns σ, M to \mathcal{A} .

\mathcal{VCO} oracle: At most q_{VC} , \mathcal{A} can make a query for the credential VCR_i corresponding to the assertion A_i . As a response, \mathcal{VCO} replies \mathcal{A} with a corresponding credential VCR_i .

\mathcal{VSO} oracle: At most q_{VS} , \mathcal{A} can make a query for the verification of a signature σ to \mathcal{VSO} with a signature σ as input. As a response, \mathcal{VSO} returns with **Accept** or **Reject** corresponding to a validation of signature σ .

Unforgeability: The unforgeability property in our model aims to provide a security against existential unforgeability under adaptive chosen message and credential exposure attack. It intentionally prevents an attacker accesses to credential queries to generate a policy-controlled signature σ_* on a new message M^* . Formally, the unforgeability provides an assurance that one, with an access to signing queries, credential queries, and the signer public parameters pk_S , should be unable to produce a policy-controlled signature on a new message M^* even with arbitrarily choosing policy POL , message M and the entire credentials $\{VCR\}$ as inputs.

We denote by $CM-A$ the adaptively chosen message and credential exposure. We also denote by $EU\mathcal{F}\text{-PCS}$ the existential unforgeability of PCS scheme. Let $\mathcal{A}_{EU\mathcal{F}\text{-PCS}}^{CM-A}$ be the adaptively chosen message and credential exposure adversary and let \mathcal{F} be a simulator. The following game between \mathcal{F} and \mathcal{A} is defined to describe the existential unforgeability of PCS scheme:

Given a choice of messages M and an access to queries \mathcal{SSO} and \mathcal{VCO} , \mathcal{A} arbitrarily make queries to oracles in an adaptive way. At the end of the above queries, we assume that \mathcal{A} outputs a forged signature σ_* on a new message M^* with respect to the public key pk_S and policy POL^* . We denote that $\overline{POL^*}$ is the entire possible set of credential of a policy POL^* . We say that \mathcal{A} wins the game if:

1. $Accept \leftarrow Verify(M^*, \sigma_*, pk_S, POL^*, \{VCR_{i,j,k}\} \in \overline{POL^*})$.
2. \mathcal{A} never made a request for a policy-controlled signature on input $M^*, pk_S, POL^*, \{VCR_{i,j,k}\} \in \overline{POL^*}$ to \mathcal{SSO} oracle.

Let $Succ_{EU\mathcal{F}\text{-PCS}}^{CM-A}(\cdot)$ be the success probability function of that $\mathcal{A}_{EU\mathcal{F}\text{-PCS}}^{CM-A}$ wins the above game.

Definition 4. We say that PCS scheme is $(\mathfrak{t}, q_H, q_{SS}, q_{VC}, \epsilon)$ -secure existentially unforgeable under a chosen message and credential exposure attack if there are no PPT adversary $\mathcal{A}_{EU\mathcal{F}\text{-PCS}}^{CM-A}$ such that the success probability $Succ_{EU\mathcal{F}\text{-PCS}}^{CM-A}(k) =$

ϵ is non-negligible in k , where $\mathcal{A}_{EUF-PCS}^{CM-A}$ runs in time at most \mathfrak{t} , make at most q_H to the random oracle, and at most q_{SS} , and q_{VC} queries to queries SSO and VCO , respectively.

Coalition-resistant: In this section, we will discuss about coalition-resistant property of PCS schemes. Coalition-resistant property aims to prevent an attacker as a group of corrupted credential holders (verifiers) to verify a policy-controlled signature σ_* on a message M^* with a policy POL which an attacker does not have enough credential to satisfy the policy POL .

The condition “verifier does not have enough credential to satisfy the policy POL ” is elaborated as follows:

At least one set of credential of assertions A in the policy $POL = \bigwedge_{i=1}^a [\bigvee_{j=1}^{a_i} [\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ is not given to the verifier such that the verifier does not have sufficient credentials to verify a policy-controlled signature on a message M with the policy POL . For example:

1. In case of $a = 1; a_i = 1; a_{i,j} > 1$, the verifier does not have one (or more) credential VCR_l of assertions $A_1, \dots, A_{a_{i,j}}$.
2. In case of $a = 1; a_i > 1; a_{i,j} > 1$, the verifier does not have one set of credential $VCR_{l,1}, \dots, VCR_{l,a_i}$ of assertions $[A_{1,j,1}, \dots, A_{1,j,a_{i,j}}]_{1 \leq j \leq a_i}$ that satisfy the first cases.
3. In the case of $a > 1; a_i > 1; a_{i,j} > 1$, the verifier has one set of credential (e.g., $[VCR_{l,j,1}, \dots, VCR_{l,j,a_{i,j}}]_{1 \leq j \leq a_i}$) of assertions $[A_{i,j,1}, \dots, A_{i,j,a_{i,j}}]_{1 \leq i \leq a, 1 \leq j \leq a_i}$ that does not satisfy the second case.

Formally, the coalition-resistant provides an assurance that one, with an access to signing queries, credential queries, and the signer public parameters pk_S , should be unable to distinguish a valid signature out of two policy-controlled signature on a message M^* even by arbitrarily choosing policy POL^* , message M^* and the entire credential $\{VCR\}$ except one set of credentials that make unsatisfiable to policy POL^* as inputs. This intentionally prevents a distinguisher to distinguish a valid signature from a (simulated) invalid signature on any message M with a new policy POL^* .

Let $CRI-PCS$ denote the existential coalition-resistant of PCS scheme. Let $\mathcal{A}_{CRI-PCS}^{CMP-A}$ be the adaptively chosen message and chosen policy attack and let \mathcal{F} be a simulator. The following experiment between \mathcal{F} and \mathcal{A} is prescribed the existential coalition-resistant of PCS scheme. The experiment is divided into two phases. We run them as follows:

1. **Phase 1:** With any adaptive strategies, \mathcal{A} arbitrarily sends requests of query to SSO , VCO queries. The queries response as per their design.
2. **Challenge:** At the end of the first phase, \mathcal{A} decides to challenge and then outputs M^* and $POL^* = \bigwedge_{i=1}^a [\bigvee_{j=1}^{a_i} [\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ such that:
 - a. On input POL^* and M^* , \mathcal{A} never issued a request for a policy-controlled signature to SSO queries.

- b. On input POL^* , \mathcal{A} can issue a request of credentials to \mathcal{VCO} queries, however, \mathcal{A} must not make sufficient requests of credentials to satisfy the policy POL^* as mentioned clearly above.

After that \mathcal{F} chooses a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$ then, on input a policy POL^* , a signer public key pk_S and a message M^* , \mathcal{F} makes a request for a policy-controlled signature to \mathcal{SSO} queries and responds \mathcal{A} with σ^* as an output from \mathcal{SSO} queries. Otherwise, on input a policy POL^* , a signer public key pk_S , a message M^* , a valid policy-controlled signature σ^* on message M^* with policy POL^* and a set of credentials $\{VCR_{i,j,k}\} \in \overline{POL^*}$, \mathcal{F} computes a (simulated) invalid policy-controlled signature σ^* and responds \mathcal{A} with σ^* .

3. **Phase 2:** In this phase, \mathcal{A} can return to *Phase 1* or *Challenge* as many as \mathcal{A} wants, on one condition, \mathcal{A} must have at least one set of challenges M^*, POL^*, σ^* such that
- On input POL^* and M^* , \mathcal{A} never issued a request for a policy-controlled signature to \mathcal{SSO} queries.
 - On input POL^* , \mathcal{A} can issue a request of credentials to \mathcal{VCO} queries, however, \mathcal{A} must not have enough credentials to satisfy the policy POL^* and to verify σ^* as mentioned clearly above.
4. **Guessing:** On the challenge M^*, POL^*, σ^* , \mathcal{A} finally outputs a guess b' . The distinguisher wins the game if $b = b'$.

Let $Succ_{CRI-PCS}^{CMP-A}(\cdot)$ be the success probability function of that $\mathcal{A}_{CRI-PCS}^{CMP-A}$ wins the above game.

Definition 5. We say that PCS scheme is $(\mathfrak{t}, q_H, q_{SS}, q_{VC}, \epsilon)$ -secure existentially coalition-resistant under a chosen message and chosen policy attack if there are no PPT distinguisher $\mathcal{A}_{CRI-PCS}^{CMP-A}$ such that the success probability $Succ_{CRI-PCS}^{CMP-A}(k) = |\Pr[b = b'] - \Pr[b \neq b']| = \epsilon$ is non-negligible in k , where $\mathcal{A}_{CRI-PCS}^{CMP-A}$ runs in time at most \mathfrak{t} , make at most q_H to the random oracle, and at most q_{SS} , and q_{VC} queries to queries \mathcal{SSO} and \mathcal{VCO} , respectively.

Invisibility: In this section, we will elaborate the invisibility property of PCS schemes. Intuitively, the invisibility property is to prevent an attacker, who does not have any credential to satisfy a policy POL , to verify a policy-controlled signature σ on a message M with respect to a policy POL . Formally, the invisibility provides an assurance that one, with an access to signing queries, verification queries and the signer public parameters pk_S , should be unable to distinguish a valid policy-controlled signature on a message M^* from an invalid one even with arbitrarily choosing policy POL^* and message M^* as inputs.

Let $INV-PCS$ denote the existential invisibility privacy of PCS scheme. Let $\mathcal{A}_{INV-PCS}^{CMP-A}$ be the adaptively chosen message and chosen policy distinguisher and let \mathcal{F} be a simulator. The following experiment between \mathcal{F} and \mathcal{A} is prescribed as the existential invisibility privacy of PCS scheme. The experiment is divided into two phases. We run them as follows:

1. **Phase 1:** With any adaptive strategies, \mathcal{A} arbitrarily sends requests of query to \mathcal{SSO} and \mathcal{VSO} . The queries response as their design.
2. **Challenge:** At the end of the first phase, \mathcal{A} decides to challenge and then outputs M^* and $POL^* = \bigwedge_{i=1}^a [\bigvee_{j=1}^{a_i} [\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ such that, on input POL^* and M^* , \mathcal{A} never issued a request for a policy-controlled signature to \mathcal{SSO} queries. After that \mathcal{F} chooses a random bit $b \stackrel{\$}{\leftarrow} \{0, 1\}$. If $b = 1$ then, on input a policy POL^* , a signer public key pk_S and a message M^* , \mathcal{F} makes a request for a policy-controlled signature to \mathcal{SSO} queries and responds \mathcal{A} with σ^* as an output from \mathcal{SSO} queries. Otherwise, on input a policy POL^* , a signer public key pk_S , a message M^* , a valid policy-controlled signature σ^* on message M^* with policy POL^* , \mathcal{F} computes a (simulated) invalid policy-controlled signature σ^* and responds \mathcal{A} with σ^* .
3. **Phase 2:** In this phase, \mathcal{A} can return to *Phase 1* or *Challenge* as many as it want. With one condition, \mathcal{A} must have at least one set of challenge M^*, POL^*, σ^* such that:
 - a. On input POL^* and M^* , \mathcal{A} never issued a request for a policy-controlled signature to \mathcal{SSO} queries.
 - b. On input POL^*, M^* and σ^* , \mathcal{A} never issued a request for the verification of the policy-controlled signature σ^* to \mathcal{VSO} queries.
4. **Guessing:** On the challenge M^*, POL^*, σ^* , \mathcal{A} finally outputs a guess b' . The distinguisher wins the game if $b = b'$.

Let $Succ_{INV-PCS}^{CMP-A}(\cdot)$ be the success probability function of that $\mathcal{A}_{INV-PCS}^{CMP-A}$ wins the above game.

Definition 6. We say that PCS scheme is $(\mathfrak{t}, q_H, q_{SS}, q_{VS}, \epsilon)$ -secure existentially invisibility under a chosen message and chosen policy attack if there are no PPT distinguisher $\mathcal{A}_{INV-PCS}^{CMP-A}$ such that the success probability $Succ_{INV-PCS}^{CMP-A}(k) = |\Pr[b = b'] - \Pr[b \neq b']| = \epsilon$ is non-negligible in k , where $\mathcal{A}_{INV-PCS}^{CMP-A}$ runs in time at most \mathfrak{t} , make at most q_H to the random oracle, and at most q_{SS} , and q_{VS} queries to \mathcal{SSO} and \mathcal{VSO} , respectively.

Theorem 1. The invisibility of of policy-controlled signature schemes implies the coalition-resistant property of policy-controlled signature schemes.

Proof. Assume that an invisibility adversary \mathcal{A}_I solves the invisibility of PCS scheme, we will show that an adversary \mathcal{A}_{CR} can solve the coalition-resistant of PCS scheme by using \mathcal{A}_I . Let \mathcal{S} be a simulator and then \mathcal{S} runs the existentially coalition-resistant game defined earlier with \mathcal{A}_{CR} . Meanwhile, \mathcal{A}_{CR} runs the existentially invisibility game defined earlier with \mathcal{A}_I . On access to the \mathcal{SSO} and the \mathcal{VCO} queries constructed by \mathcal{S} , \mathcal{A}_{CR} passes the \mathcal{SSO} queries from \mathcal{S} to \mathcal{A}_I . Then, by using \mathcal{VCO} queries from \mathcal{S} , \mathcal{A}_{CR} construct \mathcal{VSO} queries for \mathcal{A}_I by making queries to \mathcal{VCO} for credentials to verify a signature queried by \mathcal{A}_I . At the end of phase 2, \mathcal{A}_I outputs for challenge with M^* and POL^* to \mathcal{A}_{CR} . \mathcal{A}_{CR} then relays this challenge to \mathcal{S} . \mathcal{S} responses with σ^* . \mathcal{A}_{CR} returns σ^* to \mathcal{A}_I . Finally, \mathcal{A}_I outputs a decision b' and give to \mathcal{A}_{CR} . Then \mathcal{A}_{CR} returns b' to \mathcal{S} .

From the above experiment, it is clearly shown that if \mathcal{A}_I can solve the invisibility of PCS scheme, then \mathcal{A}_{CR} can solve the coalition-resistant of PCS scheme via \mathcal{A}_I . Hence, the coalition-resistant is a stronger model of the invisibility in the notion of PCS scheme. \square

4 PCS Scheme

4.1 High Level Idea

Prior to presenting our concrete construction of policy-controlled signature schemes, we will first describe our idea and intuition behind our construction for clarity. Intuitively, we can achieve a policy-controlled signature scheme by combining the idea of policy-based encryption schemes [11], a general signature scheme and a designated verifier signature scheme as follows. Firstly, we combine a designated verifier signature on a message into a policy-based ciphertext such that only a verifier, who has satisfied the policy, can verify the authenticity of the signature. This will constitute the part of policy-controlled signatures, which we refer to as “the encrypted designated verifier signature”. Then, a signature scheme is used to sign the concatenation of the policy and the encrypted designated verifier signature. The encrypted designated verifier signature and the signature from the above construction constitutes a policy-controlled signature. The purpose of the above construction is to ensure the authentication of the signer such that the verifier is convinced that the signer has actually generated this policy-controlled signature. The signature can be publicly verifiable, however, one could not be convinced that the signature is indeed associated to a message, unless one has the credentials satisfying the policy to verify the encrypted designated verifier signature. Hence, without revealing the verifier’s private information (the credentials associated to the policy), the verifier should not be able to convince other party that a signer generated the policy-controlled signature.

4.2 The Construction

In this section, we present our concrete construction of PCS schemes. Let $H_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1$; $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$; $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ be three distinct random one-way functions that map any string to group \mathbb{G}_1 and let $h : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$ be a collision-resistant hash function. We denote by \mathbb{G}_1 and \mathbb{G}_T groups of prime order p . Assume that there exists an efficient computationally bilinear mapping function \hat{e} which maps \mathbb{G}_1 to \mathbb{G}_T . The above mapping function is defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The scheme is described as follows.

Setup: On input a security parameter \mathcal{K} , a trusted third party randomly chooses a prime $p \approx \text{poly}(1^{\mathcal{K}})$. Select a random generator $g \in \mathbb{G}_1$ and a bilinear mapping function \hat{e} . Select hash functions $H_0(\cdot), H_1(\cdot), H_2(\cdot), h(\cdot)$. We denote by $\text{param} = (p, \hat{e}, g, H_0, H_1, H_2, h)$ the system parameters. Then, *Setup* returns param .

TKeyGen: On input a system parameters **param**, a trusted authority *TA* randomly generates a private key sk_{TA} and a public key pk_{TA} as follows: select two random integers $\mu, \gamma \in \mathbb{Z}_p$. Let $U = g^\mu; W = g^\gamma$ denote a public key. Therefore, *TKeyGen* returns $sk_{TA} = (\mu, \gamma)$ as a private key of the trusted authority and $pk_{TA} = (U, W)$ as a public key of the trusted authority.

SKeyGen: On input a system parameters **param** and a public key of the trusted authority pk_{TA} , a signer *S* randomly generates a private key sk_S and a public key pk_S as follows: select a random integer $x \in \mathbb{Z}_p$. Let $X = g^x; \hat{X} = W^x$ denote a public key. Therefore, *SKeyGen* returns $sk_S = x$ as a private key of the signer and $pk_S = (X, \hat{X})$ as a public key of the signer.

CreGen: Let *P* be a statement in the policy, e.g., *P* = ‘CIA agent’. An assertion *A* of *P* is computed as follows: $A = H_2(P)$. On input a system parameters **param**, the trusted authority’s public key pk_{TA} , the trusted authority’s private key sk_{TA} and a set of assertions A_1, \dots, A_n that verifier is satisfied to obtain, a trusted authority *TA* randomly generates each verifier’s credential strings $VCR_i = (CV_i, CR_i, CG_i)$ where *i* is an index of credentials as follows: *TA* randomly selects $\nu_i \in \mathbb{Z}_p^*$ and computes each credential $CV_i = U^{1/\nu_i}; CR_i = g^{(\mu\gamma)/\nu_i} A_i^\mu; CG_i = g^{\nu_i}$ and then returns $VCR_i = (CV_i, CR_i, CG_i)$ to the verifier as a credential of assertion A_i . Verifier checks the validity of VCR_i as follows:

$$\begin{aligned} \hat{e}(CR_i, g) &\stackrel{?}{=} \hat{e}(A_i, U) \hat{e}(W, CV_i), \\ \hat{e}(CV_i, CG_i) &\stackrel{?}{=} \hat{e}(U, g). \end{aligned}$$

Sign: Given **param**, pk_{TA} , sk_S , pk_S , $POL = \bigwedge_{i=1}^a [V_{j=1}^{a_i} [\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ and a message *M*, *S* computes a policy-controlled signature σ on a message *M* as follows:

$$\begin{aligned} r, t_1, \dots, t_a &\stackrel{\$}{\leftarrow} \mathbb{Z}_p, \quad t = \bigoplus_{i=1}^a t_i, \quad \sigma_1 = g^r, \\ \sigma_2 &= X^r, \quad \sigma_3 = \hat{X}^r, \\ M' &= M || \sigma_1 || \sigma_2 || \sigma_3 || t || t_1 || \dots || t_a || pk_S || pk_{TA} || POL, \end{aligned}$$

for $i = 1$ to a , for $j = 1$ to a_i :

$$\begin{aligned} \alpha_{i,j} &= h(\hat{e}((\prod_{k=1}^{a_{i,j}} A_{i,j,k})^{r \cdot x}, U) || H_0(M') || i || j), \\ R_{i,j} &= t_i \oplus h(\hat{e}(g^{\alpha_{i,j}}, H_0(M')^x) || i || j), \end{aligned}$$

Then compute

$$\begin{aligned} \check{M} &= \sigma_1 || \sigma_2 || \sigma_3 || t || t_1 || \dots || t_a || pk_S || pk_{TA} || POL || [R_{i,1} || \dots || R_{i,a_i}]_{1 \leq i \leq a}, \\ \sigma_4 &= H_1(\check{M})^x. \end{aligned}$$

The policy-controlled signature on a message *M* is

$$\sigma = (H_0(M'), \sigma_1, \sigma_2, \sigma_3, \sigma_4, [R_{i,1}, \dots, R_{i,a_i}]_{1 \leq i \leq a}).$$

Verify : Let $\{VCR_{i,j,k}\} = VCR_{1,j,1}, \dots, VCR_{a,j,a_{i,j}}$ be a set of credentials in \overline{POL} that verifier possessed. Given $pk_S, pk_{TA}, pk_V, \{VCR_{i,j,k}\} \subset \overline{POL}$, POL, σ and a message M , a verifier V first checks whether

$$\hat{e}(\sigma_2, g) \stackrel{?}{=} \hat{e}(\sigma_1, X), \hat{e}(\sigma_3, g) \stackrel{?}{=} \hat{e}(\sigma_2, W).$$

holds or not. If not, then V outputs **reject**. Otherwise, V computes as follows: for $i = 1$ to a :

$$\alpha_{i,j} = h\left(\left(\prod_{k=1}^{\alpha_{i,j}} (\hat{e}(CR_{i,j,k}, \sigma_2) \hat{e}(CV_{i,j,k}, \sigma_3)^{-1})\right) \| H_0(M') \| i \| j\right)$$

$$\hat{t}_i = R_{i,j} \oplus h(\hat{e}(H_0(M')^{\alpha_{i,j}}, X) \| i \| j).$$

After that, compute

$$\hat{t} = \bigoplus_{i=1}^a \hat{t}_i, \hat{M} = M \| \sigma_1 \| \sigma_2 \| \sigma_3 \| \hat{t} \| \hat{t}_1 \| \dots \| \hat{t}_a \| pk_S \| pk_{TA} \| POL.$$

Then, V checks whether $H_0(\hat{M}) \stackrel{?}{=} H_0(M')$, $\hat{e}(\sigma_4, g) \stackrel{?}{=} \hat{e}(H_1(\sigma_1 \| \sigma_2 \| \sigma_3 \| \hat{t} \| \hat{t}_1 \| \dots \| \hat{t}_a \| pk_S \| pk_{TA} \| POL) \| [R_{i,1} \| \dots \| R_{i,a_i}]_{1 \leq i \leq a}, X)$ holds or not. If not, then it outputs **reject**. Otherwise, it outputs **accept**.

5 Security Analysis

Theorem 2. *Our policy-controlled signature scheme is existentially unforgeable under an adaptive chosen message and credential exposure attack if the CDH assumption holds in the random oracle model.*

Theorem 3. *In the random oracle model, the proposed policy-controlled signature scheme is existentially coalition-resistant against adaptively chosen message and chosen policy attack $\mathcal{A}_{CRI-PC}^{CMP-A}$ attack if the DBDH assumption is hold.*

Due to the page limitation, please find the proof for Theorem 2 and Theorem 3 in the full version of this paper [28].

6 Implementation

In this section, we briefly discussed our implementation to realize policy-controlled signatures. We incorporated the P3P privacy policy description language [27] and extended the description to capture the notion of policy-controlled signatures. We augment the XML digital signature with the P3P privacy policy to ensure that the policies attached will be enforced. An example of a policy-controlled signature given in our earlier scenario is provided in Figure 1. We note that in this section, we only demonstrate how the policy can be written in the P3P description language and we do not describe its enforcement, as it is not in the scope of this paper. The purpose of this section is just merely to demonstrate that our primitive is also applicable in practice.

```

<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
. . .
<SignatureValue>NC3E~LE=</SignatureValue>
<KeyInfo>
<POLICIES>
  <POLICY discuri="http://p3pbook.com/privacy.html" name="policy">
    <RECIPIENT>
      <DATA ref='‘CIA Agent’' />
      <DATA ref='‘KGB Agent’' />
      <DATA ref='‘Agents’' AND ref='‘Authorized by US’' />
    </RECIPIENT>
  </POLICY>
</POLICIES>

```

Fig. 1. P3P for policy-controlled signatures

7 Conclusion

In this paper, we introduced the notion of policy-controlled signatures that allows a signer to control the verification of his signature by attaching policies. If the verifier satisfies the policies provided, then the verifier can test the authenticity of the message. Otherwise, the verifier cannot do so. Only a verifier who has credentials satisfying the policies provided by the signer can verify the policy-controlled signatures. We argue that this cryptographic primitive has many applications, in particular the ones that involve sensitive information. We presented a security model for PCS schemes, together with a concrete construction that is secure in our model.

References

1. Bagga, W., Molva, R.: Policy-based cryptography and applications. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 72–87. Springer, Heidelberg (2005)
2. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
3. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.: Secret Handshakes from Pairing-based Key Agreements. In: 2003 IEEE Symposium on Security and Privacy, pp. 180–196 (2003)
4. Diffie, W., Hellman, M.E.: New directions in cryptography IT-22(6), 644–654 (November 1976)
5. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks 17(2), 281–308 (April 1988); Special issue on cryptography
6. Laguillaumie, F., Vergnaud, D.: Multi-designated verifiers signatures: anonymity without encryption. Inf. Process. Lett. 102(2-3), 127–132 (2007)

7. Laguillaumie, F., Vergnaud, D.: Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 105–119. Springer, Heidelberg (2005)
8. Li, Y., Lipmaa, H., Pei, D.: On delegatability of four designated verifier signatures. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 61–71. Springer, Heidelberg (2005)
9. Lipmaa, H., Wang, G., Bao, F.: Designated verifier signature schemes: Attacks, new security notions and a new construction. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 459–471. Springer, Heidelberg (2005)
10. Huang, X., Mu, Y., Susilo, W., Zhang, F.: Short designated verifier proxy signature from pairings. In: Enokido, T., Yan, L., Xiao, B., Kim, D.Y., Dai, Y.-S., Yang, L.T. (eds.) EUC-WS 2005. LNCS, vol. 3823, pp. 835–844. Springer, Heidelberg (2005)
11. Susilo, W., Zhang, F., Mu, Y.: Identity-based strong designated verifier signature schemes. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 313–324. Springer, Heidelberg (2004)
12. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Lai, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (2003)
13. Laguillaumie, F., Libert, B., Quisquater, J.-J.: Universal designated verifier signatures without random oracles or non-black box assumptions. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 63–77. Springer, Heidelberg (2006)
14. Bagga, W., Molva, R.: Collusion-free policy-based encryption. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 233–245. Springer, Heidelberg (2006)
15. Bagga, W., Crosta, S., Molva, R.: Proof-carrying proxy certificates. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 321–335. Springer, Heidelberg (2006)
16. Bagga, W., Crosta, S., Michiardi, P., Molva, R.: Establishment of ad-hoc communities through policy-based cryptography. *Electr. Notes Theor. Comput. Sci.* 171(1), 107–120 (2007)
17. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
18. Bresson, E., Stern, J., Szydło, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002)
19. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 384–398. Springer, Heidelberg (2004)
20. Herranz, J., Sáez, G.: Forking lemmas for ring signature schemes. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 266–279. Springer, Heidelberg (2003)
21. Zhang, F., Kim, K.: Id-based blind signature and ring signature from pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002)
22. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006)
23. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 166–180. Springer, Heidelberg (2007)

24. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 181–200. Springer, Heidelberg (2007)
25. Liu, J.K., Wong, D.S.: On the security models of (threshold) ring signature schemes. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 204–217. Springer, Heidelberg (2005)
26. Liu, J.K., Wei, V.K., Wong, D.S.: A separable threshold ring signature scheme. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 12–26. Springer, Heidelberg (2004)
27. W3C: Platform for privacy preferences (p3p) project, <http://www.w3.org/P3P/>
28. Thorncharoensri, P., Susilo, W., Mu, Y.: Policy-controlled signatures (full version). can be obtained from the first author (2009)

Public Key Encryption without Random Oracle Made Truly Practical

Puwen Wei^{1,*}, Xiaoyun Wang^{1,2,*}, and Yuliang Zheng³

¹ Key Laboratory of Cryptologic Technology and Information Security,
Ministry of Education, Shandong University, Jinan 250100, China

weipuwen@mail.sdu.edu.cn

² Institute for Advanced Study, Tsinghua University, Beijing 100084, China

xiaoyunwang@mail.tsinghua.edu.cn

³ Department of Software and Information Systems,

University of North Carolina at Charlotte, Charlotte, NC 28223, USA

yzheng@uncc.edu

Abstract. An important research area in the past decade is to search for efficient cryptographic schemes that do not rely for their security on the controversial random oracle assumption. In this paper, we continue this line of endeavors and report our success in identifying a very efficient public key encryption scheme whose formal security proof does not require a random oracle. Specifically, we show how to modify a universal hash based public key encryption scheme proposed by Zheng and Seberry at Crypto'92, in such a way that the resultant scheme not only preserves efficiency but also admits provable security against adaptive chosen ciphertext attack without a random oracle. We also compare the modified Zheng-Seberry scheme with related encryption schemes in terms of efficiency and underlying assumptions, supporting our conclusion that the modified Zheng-Seberry scheme is preferable to its competitors.

Keywords: random oracle, universal hash, public key encryption.

1 Introduction

The notion of chosen ciphertext security was introduced by Naor and Yung [24]. Then, Rackoff and Simon [26] provided a stronger notion called indistinguishability under adaptive chosen ciphertext attack (IND-CCA2). Adaptive chosen ciphertext security has since become a standard notion for the security of public key encryption. A considerable amount of work on the construction of adaptive chosen ciphertext secure encryption have been presented. Some of these research results are based on non-interactive zero knowledge proofs [13], which are not quite practical in real world applications. To construct an efficient encryption scheme, many encryption techniques have been proposed in the so-called random oracle model [4] [14] [3]. The random oracle model, however, is one of the most

* Supported by the National Natural Science Foundation of China (NSFC Grant No. 60525201) and 973 Project (No.2007CB807902).

controversial issues in cryptography. A notable argument against the random oracle model was made by Canetti, Goldreich and Halevi [8] who demonstrated that there exist cryptographic schemes that are secure in the random oracle model but insecure for any instantiation of the random oracle. Recently, Leurent and Nguyen [23] showed that instantiations of full domain hash functions (random oracles) proposed in the literature are insecure. They also advocated to assess carefully the impact of potential flaws in random oracle instantiations on a system that relies on the instantiations.

To address the concern over random oracles, an obvious approach is to design an adaptive chosen ciphertext secure public key encryption scheme that does not rely on a random oracle. The often cited adaptive chosen ciphertext secure encryption scheme proposed by Cramer and Shoup [10] represents the first concrete result in this line of research. A multiple number of techniques have since been proposed and studied by many researchers. Most of these techniques, however, share a common drawback that impedes their possible adoption in practice, that is, they generally require at least a few times more computation than their random oracle based counterparts.

Given the computational superiority of random oracle based encryption, it is a shared view amongst most researchers that alternative encryption techniques without random oracles will not be able to win over practitioners unless these alternatives afford a computational speed comparable to that enjoyed by random oracle based techniques.

A major advantage of random oracle based schemes [4] [14] [3] lies in its simplicity. To preserve the simplicity while not relying on a random oracle for security proofs, new computational assumptions have been examined. One such effort is made by Pandey, Pass and Vaikuntanathan [25] who introduce a few complexity theoretical hardness assumptions that abstract out concrete properties of a random oracle. Based on these assumptions, they are able to solve a number of open problems, including the construction of a non-interactive concurrently non-malleable string commitment. Their results point to an interesting approach towards designing efficient and provably secure cryptographic schemes without random oracles. We note that although these assumptions are stronger than traditional cryptographic hardness assumptions, they seem quite reasonable and it is conceivable that, like many other assumptions in the field such as the decisional Diffie-Hellman assumption, this type of new assumptions may gain wider acceptance after further screening by peers in the field.

1.1 Our Contribution

The goal of this paper is to search for a public key encryption scheme that (1) does not rely on a random oracle, (2) is adaptive chosen ciphertext secure, and (3) is truly practical in that it requires no more exponentiations of large integers than does a comparable random oracle based scheme. To achieve our goal, we examine a variant of Pandey et al.'s assumption [25], called the adaptive DDH assumption. Based on the adaptive DDH assumption, a modified version of

Zheng and Seberry’s encryption scheme proposed in [30] is proved to be adaptive chosen ciphertext secure without a random oracle.

Zheng and Seberry [30] proposed three simple methods for immunizing public key cryptosystems against chosen ciphertext attacks. The nature of the three methods is the same. They immunized a public key cryptosystem by appending to each ciphertext a tag that is correlated to the message to be encrypted. Soldara, Seberry and Qu [29] showed the insecurity of the first scheme, denoted by $\text{Zheng-Seberry}_{1wh}$, on some special circumstances and attempted to modify $\text{Zheng-Seberry}_{1wh}$ resulted on an El Gamal variant. Based on the Gap Diffie-Hellman assumption (GDH), Baek and Zheng [2] provided a security proof for the slightly modified version of $\text{Zheng-Seberry}_{1wh}$, in the random oracle model, leaving as an open problem proofs for the other two schemes. The focus of this paper is to modify the second scheme in [30], denoted by $\text{Zheng-Seberry}_{uh}$, so that the resultant scheme is adaptive chosen ciphertext secure (see Section 4). The scheme $\text{Zheng-Seberry}_{uh}$ is worth studying for the following reasons: First, the scheme immunizes public key encryption against adaptive chosen ciphertext attacks with the help of a universal hash function. This allows the scheme to steer clear of a one-way hash function with non standard output size, whereby successfully averting potential risks recently discovered in [23]. Second, the input length of a plaintext can be arbitrary, while the overhead of the corresponding ciphertext is a constant. As a result, the ratio between the length of the ciphertext and that of the plaintext can be close to 1 as the length of the plaintext increases.

1.2 Related Work

Hybrid encryption, which is also known as the KEM-DEM approach, applies a public key cryptosystem to encapsulate the key of a symmetric cryptosystem and the symmetric cryptosystem is subsequently used to conceal data. Cramer and Shoup first generalized the notion in their work [27][11]. Kurosawa and Desmedt [22] later presented a more efficient hybrid encryption scheme by using a KEM which is not necessarily adaptive chosen ciphertext secure. More recently, Kiltz et al. [20] improved on the Kurosawa-Desmedt technique and proposed a new approach to design adaptive chosen ciphertext secure hybrid encryption schemes without a random oracle. Compared with Kiltz et al.’s concrete scheme [20] which relies on the DDH assumption and AE-OT¹ secure symmetric encryption, our modified $\text{Zheng-Seberry}_{uh}$ scheme is conceptually much simpler and relies only on the adaptive DDH assumption. More important, this newly modified scheme requires less computation time than Kiltz et al.’s.

Another important progress was made by Hofheinz and Kiltz [18] recently. They proposed a new public key encryption scheme based on factoring. Their scheme requires only roughly two exponentiations in encryption and roughly one exponentiation in decryption. (Here, “roughly” two or one exponentiation means

¹ According to [20], a symmetric cipher is AE-OT secure if it satisfies (one-time) ciphertext indistinguishability (IND-OT) and (one-time) ciphertext integrity (INT-OT).

two or one full exponentiation and additional exponentiations with small exponents.) While for the encryption schemes based on discrete logarithm, DHIES [1] is one of the most efficient schemes without random oracle. Compared with DHIES which relies on the Oracle Diffie-Hellman (ODH) assumption together with the security of symmetric encryption and a message authentication code, our modified scheme relies only on the adaptive DDH assumption and preserves the computational efficiency of Zheng-Seberry_{uh}. However, it is fair to say that our modified Zheng-Seberry scheme and DHIES are comparable, each having its own pros and cons in practice. With DHIES, all three assumptions on symmetric encryption, MAC and ODH are responsible for the security of DHIES and it is relatively easy to select proper candidates to realize each function of the assumption. With our modified Zheng-Seberry scheme, the adaptive DDH assumption which is solely responsible for the security of the scheme is slightly stronger than the ODH assumption required by DHIES.

2 Preliminaries

Notation and Definition. $|X|$ denotes the length of a binary string X or the size of (or number of elements in) a set X . $x \stackrel{R}{\leftarrow} X$ denotes picking an element x from X uniformly at random. $x \leftarrow A(x_1)$ denotes the experiment of running an algorithm A on input x_1 and outputting x . $x||y$ denotes the concatenation of strings x and y . A function $\mu : \mathbb{N} \rightarrow \mathbb{R}$ is called negligible in n if for every positive polynomial $p(\cdot)$ and all sufficiently large n 's, we have $\mu(n) < 1/p(n)$.

Universal hashing [9]. A family of functions $H : \{0, 1\}^P \rightarrow \{0, 1\}^l$ is a universal family of hash functions if, for every $x_1 \neq x_2 \in \{0, 1\}^P$ and every $y_1, y_2 \in \{0, 1\}^l$, the number of functions in H mapping x_1 to y_1 and x_2 to y_2 is precisely $|H|/2^{2l}$, where $|H|$ denotes the number of functions in H .

For the security proof in this paper, we need the following lemma whose proof can be found in [28].

Lemma 1. [28] *Let S_1, S_2 , and S' be events defined on a probability space such that $\Pr[S_1 \wedge \neg S'] = \Pr[S_2 \wedge \neg S']$. Then we have $|\Pr[S_1] - \Pr[S_2]| \leq \Pr[S']$*

3 New Assumptions

In this section, we give the definitions of the adaptive DDH assumption and other related assumptions. First, we recall the definition of an adaptive one-to-one one-way function introduced in [25]. In the definition, an adversary picks an index tag^* and is given $y^* = f_{tag^*}(x^*)$ for a random x^* in the domain of $f_{tag^*}(x)$. The aim of the adversary is to compute x^* . The difference between the traditional definition for an one-way function and the one in [25] is that the adversary in [25] has access to a ‘‘magic oracle’’ $\mathcal{O}_{tag}(\cdot, \cdot)$ that on input (tag, y) with $tag \neq tag^*$, returns $f_{tag}^{-1}(y)$. The security requirement is that the adversary

can compute x^* only with a negligible probability, even if the adversary can get help from the “magic oracle”. Similarly to the definition of adaptive one-way function, the definition of adaptive pseudorandom generator G_{tag} in [25] requires that the adversary can not tell the output of G_{tag^*} from the random string, even if the adversary can get help from a magic oracle that, on input (tag, y) with $tag \neq tag^*$, returns 0 or 1 depending on whether y is in the range of G_{tag} or not. Formal definitions of the adaptive one-way function and the adaptive pseudorandom generator (PRG) in [25] are given in Appendix.

Combining definitions of the adaptive one-way function and the adaptive PRG, we have the definition for a variant of the adaptive PRG. The variant is similar to the definition of the adaptive PRG except that, the adversary A has some auxiliary information $f_{tag}(x)$ on a seed x and interacts with the oracle $\mathcal{O}_{tag}(\cdot, \cdot, \cdot)$.

Definition 1. (*Auxiliary adaptive PRG*) *Let*

$$\mathcal{G} = \{G_{tag} : \{0, 1\}^n \rightarrow \{0, 1\}^{s(n)}\}_{tag \in \{0, 1\}^n}$$

be a pseudorandom generator (PRG). And let $\mathcal{O}_{tag}(\cdot, \cdot, \cdot)$ denote an oracle that, on input $(tag', f_{tag'}(x), y)$ such that $tag' \neq tag$, $|tag'| = |tag|$, outputs the seed x if

- $y = G_{tag'}(x)$, and
- x is consistent with its auxiliary information $f_{tag'}(x)$.

$\mathcal{O}_{tag}(\cdot, \cdot, \cdot)$ *outputs \perp otherwise.*

We say that the PRG \mathcal{G} is adaptively secure if, for any probability polynomial-time adversary A which has the auxiliary information $f_{tag}(x)$ on the seed x and has access to the oracle $\mathcal{O}_{tag}(\cdot, \cdot, \cdot)$, there exists a negligible function μ such that for all n and for all tags $tag \in \{0, 1\}^n$,

$$|Adv_A^{real} - Adv_A^{rand}| \leq \mu(n)$$

where Adv_A^{real} denotes $\Pr[x \leftarrow U_n : A^{\mathcal{O}_{tag}(\cdot, \cdot, \cdot)}(f_{tag}(x), G_{tag}(x)) = 1]$, Adv_A^{rand} denotes $\Pr[y \leftarrow U_{s(n)} : A^{\mathcal{O}_{tag}(\cdot, \cdot, \cdot)}(f_{tag}(x), y) = 1]$ and the probability is over the random choice of y and x , and the coin-tosses of A .

Definition 1 is a combination of definitions of the adaptive one-way function and the adaptive PRG in that the auxiliary information on x in Definition 1 can be replaced by an one-way function $f(x)$ and the inversion oracle $\mathcal{O}_{tag}(\cdot, \cdot, \cdot)$ plays the role of $\mathcal{O}_{tag}(\cdot, \cdot)$ in the definition of adaptive one-way function. In addition, Definition 1 implies that the adversary can not invert the one-way function $f(x)$ even with the help from $\mathcal{O}_{tag}(\cdot, \cdot, \cdot)$. A candidate construction for an auxiliary adaptive PRG, based on AES, is defined by $G_{tag}(x) = AES_x(tag||0)||AES_x(tag||1)$.

From Definition 1 and the specific number theory assumption DDH, we derive the definition of the adaptive DDH assumption.

Let \mathcal{G} be a group with prime order q . $g \in \mathcal{G}$ is the generator. $G_{tag}(\cdot) : \mathcal{G} \rightarrow \{0, 1\}^*$ is a pseudorandom generator. $G_{tag}(\cdot)_{[i, \dots, j]}$ denotes the substring from the i -th bit to the j -th bit of the output of $G_{tag}(\cdot)$.

Definition 2. (*Adaptive DDH assumption*) Given

$$\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+W]}\}$$

where $a, b, c \in Z_q$, it is computationally infeasible for any PPT distinguisher D to tell whether $c = ab$, even if D has access to an oracle $\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)$, where P and W are polynomials in a security parameter.

The oracle $\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)$, on input $(g^{a'}, g^{b'}, G_{g^{a'}, g^{b'}}(g^{c'})_{[P+1, \dots, P+W]})$, outputs $g^{a'b'}$ if the input $(g^{a'}, g^{b'}, G_{g^{a'}, g^{b'}}(g^{c'})_{[P+1, \dots, P+W]})$ satisfies:

- $(g^{a'}, g^{b'}, G_{g^{a'}, g^{b'}}(g^{c'})_{[P+1, \dots, P+W]}) \neq (g^a, g^b, G_{g^a, g^b}(g^c)_{[P+1, \dots, P+W]})$
- $G_{g^{a'}, g^{b'}}(g^{a'b'})_{[P+1, \dots, P+W]} = G_{g^{a'}, g^{b'}}(g^{c'})_{[P+1, \dots, P+W]}$

Otherwise, the oracle outputs \perp .

That is, for all PPT D , there is a negligible function μ such that

$$\left| \Pr_{a, b, c \stackrel{R}{\leftarrow} Z_q} [D^{\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)}(S) = 1] - \Pr_{a, b \stackrel{R}{\leftarrow} Z_q} [D^{\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)}(S') = 1] \right| \leq \mu(n)$$

where $S = (g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+W]})$, $S' = (g, g^a, g^b, G_{g^a, g^b}(g^{ab})_{[1, \dots, P+W]})$, and n is the security parameter. A quadruple $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+W]}\}$ satisfying $c = ab$ is called an adaptive DDH quadruple.

Remark. Comparing with Definition [1](#), (g^a, g^b) is not only a tag but also represents some auxiliary information on g^{ab} . Note that it is not required that the length of the substring of $G_{g^{a'}, g^{b'}}(g^{c'})$ in the adversary's query be equal to that of $G_{g^a, g^b}(g^c)_{[1, \dots, P+W]}$. However, the length of $G_{g^{a'}, g^{b'}}(g^{c'})_{[P+1, \dots, P+W]}$, that is W , should be large enough to guarantee that the adversary can guess a “right” query only with a negligible probability. Intuitively, that means, in almost all cases, the oracle does not provide any “useful” help for the adversary. However that does not mean the adversary can not provide the right query with a non-negligible probability. In fact, the adversary can randomly pick a', b' and generate the “right” query $(g^{a'}, g^{b'}, G_{g^{a'}, g^{b'}}(g^{a'b'})_{[P+1, \dots, P+W]})$ by himself. Although the oracle's answer to such a query does not provide any useful information for the adversary, it is important for the simulation in the security proof, which will be explained later.

3.1 Relationships with Other Assumptions

HDH, ODH and SDH assumptions. Abdalla et al. introduce three related notions, which are the hash Diffie-Hellman assumption (HDH), the oracle Diffie-Hellman (ODH) assumption and the strong Diffie-Hellman assumption (SDH) [\[5\]](#) [\[1\]](#). It seems that the ODH assumption and the adaptive DDH assumption are similar in flavor. But the adversary's power in the adaptive DDH assumption is much more restricted, as the adversary can get the help of the oracle only if it can produce a useful and “right” query, which happens with only a negligible probability.

Non-malleable pseudorandom generator. In order to prove the security (\$NM\$-CPA) of OAEP without random oracle, Boldyreva and Fischlin [7] fully instantiated OAEP by assuming special properties of the two pseudorandom generators G and H in OAEP. To be more precise, G is a near-collision resistant trapdoor pseudorandom generator, which can recover the pre-image s of $G(s)$ according to the k least significant bits of $G(s)$; H is a non-malleable pseudorandom generator. Our adaptive DDH assumption is closely related to their assumption. To some extent, the adaptive DDH combines the above properties of G and H , and takes advantage of concrete algebra structures to replace the random oracle.

4 Modified Zheng-Seberry_{uh} Scheme

First, we give the description of the modified Zheng-Seberry_{uh} in Table 1. Assume that $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ is a family of universal hash functions. Each function in H is specified by a string of exactly Q bits. Denote by h_s the function in H that is specified by a string $s \in \{0, 1\}^Q$. L denotes an encryption label, which consists of public data. In addition, m denotes a plaintext to be encrypted. Our major modification to the original Zheng-Seberry_{uh} scheme [30] is to increase the output length of the pseudorandom generator by W bits. These additional W bits play the role of a tag for an ephemeral key y_A^x and will be

Table 1. The modified Zheng-Seberry_{uh} Scheme

Modified Zheng-Seberry _{uh} Scheme	
<p>Public parameters: A label L, a universal class of hash functions $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$, a group \mathcal{G}, a generator g of \mathcal{G} with order q, and an adaptively secure pseudorandom generator $G_{tag} : \mathcal{G} \rightarrow \{0, 1\}^*$.</p> <p>Key Generation: Choose x_A randomly in \mathbb{Z}_q^* and compute $y_A = g^{x_A}$. The public key is y_A and the private key is x_A.</p>	<p>Encryption $E_{uhf}(y_A, m, L)$</p> <ol style="list-style-type: none"> 1. $x \xleftarrow{R} \mathbb{Z}_q^*$, $r = y_A^x$ 2. $c_1 = g^x$. Let $tag = (y_A, c_1)$. 3. $s = G_{tag}(r)_{[1, \dots, Q]}$, $t = h_s(m L)$ 4. $z = G_{tag}(r)_{[Q+1, \dots, Q+P+W]}$, $c_2 = z \oplus (m t 0^W)$ <p>Output the ciphertext (c_1, c_2).</p>
	<p>Decryption $D_{uhf}(x_A, y_A, c_1, c_2, L)$</p> <ol style="list-style-type: none"> 1. $r' = c_1^{x_A}$, $s' = G_{tag}(r')_{[1, \dots, Q]}$, $z' = G_{tag}(r')_{[Q+1, \dots, Q+P+W]}$, 2. $m' t' = (c_2 \oplus z')_{[1, \dots, P]}$, where $m' = (c_2 \oplus z')_{[1, \dots, P-l]}$, $t' = (c_2 \oplus z')_{[P-l+1, \dots, P]}$ 3. if $h_{s'}(m' L) = t'$ and $z'_{[P+1, \dots, P+W]} = c_2_{[P+1, \dots, P+W]}$, then output m' as a plaintext; otherwise output \perp.

sent to a recipient as part of a ciphertext. In practice, in order to minimize the impact of these additional bits on the efficiency of the scheme, W should be chosen to be as short as practical. For a security level of 2^{80} , we suggest $W \geq 160$. Additionally, the pseudorandom generator $G(\cdot)$ is required to be a adaptively secure pseudorandom generator $G_{tag}(\cdot)$, where $tag = (y_A, c_1)$.

Other modifications. A public label L is employed in Table [1](#). Using such a label is a widely adopted practice and does not affect the security proof. Besides, the universal hash value is encrypted together with a message, which allows the use of a broader range of universal hash functions that may not necessarily hide all the information on a message.

4.1 Security Proof of the Modified Zheng-Seberry_{uh} Scheme

Theorem 1. Assuming the adaptive DDH assumption holds, the modified Zheng-Seberry_{uh} scheme is secure against adaptive chosen ciphertext attacks.

Proof. The main idea of the security proof is to construct three adaptive chosen ciphertext attack games, which are denoted by Game 1, Game 2 and Game 3, and prove that the adversary's views in these games are indistinguishable.

Game 1: Game 1 is a real run of a standard adaptive chosen ciphertext attack game. After the adversary submits a pair of plaintexts (m_0, m_1) in the challenge phase, the challenger creates a target ciphertext as follows: $c^* = (c_1^*, c_2^*) = (g^{x^*}, z^* \oplus (m_\beta || t^* || 0^W))$, where $t^* = h_{s^*}(m_\beta || L)$, and $\beta \xleftarrow{R} \{0, 1\}$.

Game 2: Game 2 is similar to Game 1 except that the target ciphertext is modified to $c_+^{**} = (g^{x^*}, z^{**} \oplus (m_\beta || t^{**} || 0^W))$, where $s^{**} = G_{y_A, g^{x^*}}(r^{**})_{[1, \dots, Q]}$, $z^{**} = G_{y_A, g^{x^*}}(r^{**})_{[Q+1, \dots, Q+P+W]}$, $r^{**} \xleftarrow{R} \mathcal{G}$, $t^{**} = h_{s^{**}}(m_\beta || L)$.

Game 3: Game 3 is similar to Game 2 except that the target ciphertext is modified to $c_+^* = (g^{x^*}, u_3 \oplus (m_\beta || t_+^* || 0^W))$, where $u_2 \xleftarrow{R} \{0, 1\}^Q$, $u_3 \xleftarrow{R} \{0, 1\}^{P+W}$, $t_+^* = h_{u_2}(m_\beta || L)$. Since the distribution of c_+^* is independent of the choice of β , the probability that the adversary can guess β correctly in Game 3 is $1/2$. That is $\Pr[\text{Game 3}] = 1/2$, where $\Pr[\text{Game } i]$ denotes the probability that the adversary wins Game i , for $1 \leq i \leq 3$.

Next, we will prove that $|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \leq \mu(k)$, where $\mu(k)$ is a negligible function. Assume for contradiction that there exists a polynomial $p(k)$ such that, for infinitely many k 's, $|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \geq 1/p(k)$, which means there exists an adversary B for Game 1 and Game 2 such that $|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]|$ is non-negligible. We show how to construct a PPT algorithm A to break the adaptive DDH assumption using B , by explicitly constructing an experiment of statistical test for the adaptive DDH problem.

Given $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, Q+P+W]}\}$, A sets $y_A = g^a$ and simulates the adaptive chosen ciphertext attack game for the adversary B in the following experiment.

Experiment: A sets the target ciphertext (c_1^*, c_2^*) to

$$(g^b, G_{g^a, g^b}(g^c)_{[Q+1, \dots, Q+P+W]} \oplus (m_\beta \| h_{G_{g^a, g^b}(g^c)_{[1, \dots, Q]}}(m_\beta \| L) \| 0^W))$$

and uses the oracle $\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)$ to answer the decryption query. Notice that the oracle $\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)$ would output \perp if the challenger does not propose the “right” query. More precisely, when the challenger receives the decryption query (c_1, c_2) , he computes $T = c_{2[P+1, \dots, P+W]}$ and decrypts as follows

1. If $c_1 \neq c_1^*$, the challenger makes the query (g^a, c_1, T) to the oracle $\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)$.
 - If the oracle returns the answer r , the challenger can compute

$$\begin{aligned} m \| t &= c_{2[1, \dots, P]} \oplus G_{g^a, g^b}(r)_{[Q+1, \dots, Q+P]} \\ s &= G_{g^a, g^b}(r)_{[1, \dots, Q]} \end{aligned}$$

and check whether $t = h_s(m \| L)$. If $t = h_s(m \| L)$, the challenger returns the plaintext m . Otherwise, the challenger outputs \perp .

- If the oracle outputs \perp which means the (g, g^a, c_1, T) is not an adaptive DDH quadruple and the corresponding ciphertext is not valid, then the challenger outputs \perp .
2. If $c_1 = c_1^*, c_2 \neq c_2^*, T = T^*$, where $T^* = c_{2[P+1, \dots, P+W]}^*$, the challenger can not get help from the oracle $\mathcal{O}_{g^a, g^b}(\cdot, \cdot, \cdot)$ and outputs \perp . Let $\Pr[Bad]$ denote the probability that (c_1, c_2) is a valid ciphertext such that $c_1 = c_1^*, c_{2[1, \dots, P-l]} \neq c_{2[1, \dots, P-l]}^*, T = T^*$. $\Pr[Bad]$ is negligible, because the adversary needs to find a c_2 satisfying

$$\begin{aligned} (c_2 \oplus z^*)_{[P-l+1, \dots, P]} &= h_{s^*}((c_2 \oplus z^*)_{[1, \dots, P-l]} \| L) \\ (c_2^* \oplus z^*)_{[P-l+1, \dots, P]} &= h_{s^*}((c_2^* \oplus z^*)_{[1, \dots, P-l]} \| L) \end{aligned}$$

According to the definition of the universal hash functions, if h is chosen uniformly from the universal class H , for every $c_{2[1, \dots, P]}, c_{2[1, \dots, P]}^* \in \{0, 1\}^P$ with $c_{2[1, \dots, P]} \neq c_{2[1, \dots, P]}^*, c_{2[P-l+1, \dots, P]}$ and $c_{2[P-l+1, \dots, P]}^*$ are uniformly and independently distributed over $\{0, 1\}^l \times \{0, 1\}^l$. Therefore, the adversary can find such a c_2 only with negligible probability $1/2^l$. Otherwise, it would imply that h is not chosen uniformly from H . That means, the pseudorandom string s^* could be distinguished from a random string by an efficient algorithm with a non-negligible advantage. This is a contradiction.

3. Otherwise, the challenger outputs \perp .

Let $\Pr[Exp]$ denote the probability that the adversary B wins the above game in the experiment. The following claims, Claim 1 and Claim 2, show that, if $|\Pr[Game 1] - \Pr[Game 2]|$ is non-negligible, then whether $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+Q+W]}\}$ is an adaptive DDH quadruple or not can be decided with a non-negligible advantage. Due to Claim 1 and Claim 2, we have Claim 3. More details of proofs of Claim 1, Claim 2 and Claim 3 are given in Appendix.

Claim 1. If $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+Q+W]}\}$ is an adaptive DDH quadruple, then $|\Pr[\text{Game 1}] - \Pr[\text{Exp}]|$ is negligible and $|\Pr[\text{Game 2}] - \Pr[\text{Exp}]|$ is non-negligible.

Claim 2. If $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+Q+W]}\}$ is not an adaptive DDH quadruple, then $|\Pr[\text{Game 2}] - \Pr[\text{Exp}]|$ is negligible and $|\Pr[\text{Game 1}] - \Pr[\text{Exp}]|$ is non-negligible.

Claim 3. $|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \leq \mu(k)$ if the adaptive DDH assumption holds, where $\mu(k)$ is a negligible function.

Finally, $|\Pr[\text{Game 3}] - \Pr[\text{Game 2}]|$ is also negligible if G is a secure pseudo-random generator. (Otherwise, Game 2 would serve as an efficient algorithm to distinguish the output distribution of G from the uniform distribution.) Hence, we obtain the following claim:

Claim 4. $|\Pr[\text{Game 3}] - \Pr[\text{Game 2}]| \leq \mu'(k)$ if G_{tag} is a secure pseudorandom generator, where $\mu'(k)$ is a negligible function.

From Claim 3 and Claim 4, we have

$$\begin{aligned} |\Pr[\text{Game 1}] - 1/2| &= |\Pr[\text{Game 1}] - \Pr[\text{Game 3}]| \\ &\leq |\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| + \\ &\quad |\Pr[\text{Game 2}] - \Pr[\text{Game 3}]| \\ &\leq \mu(k) + \mu'(k) \end{aligned}$$

where $\mu(k) + \mu'(k)$ is a negligible function.

That is, the adversary can win the standard adaptive chosen ciphertext attack game with only a negligible advantage. This completes the proof of Theorem 1.

5 Instantiation

First we note that an ϵ -AXU hash function [12] can be used in place of a universal hash function. One may also use an efficient universal hash function family proposed by Bernstein [6]. Such a substitution almost does not affect the security proof. In fact, only minor revisions need to be made in the security proofs. Specifically, in Case 2 of the experiment for the security proof of the modified Zheng-Seberry_{uh} scheme, the probability that the adversary can find c_2 satisfying $(c_2 \oplus z^*)_{[P-l+1, \dots, P]} = h_{s^*}((c_2 \oplus z^*)_{[1, \dots, P-l]} || L)$ and $(c_2^* \oplus z^*)_{[P-l+1, \dots, P]} = h_{s^*}((c_2^* \oplus z^*)_{[1, \dots, P-l]} || L)$ needs to be changed. To instantiate the adaptively secure pseudorandom generator $G_{\text{tag}}(\cdot)$, we can use the HMAC-based key derivation function (KDF) [21], which follows the extract-then-expand paradigm.

6 Comparison

For the modified Zheng-Seberry_{uh} scheme, the length of a ciphertext is $|m| + |p| + 320$, where $|p|$ denotes the binary length of an element in \mathcal{G} . Thanks to the use of the pseudorandom generator and the universal hash function, the input length of the plaintext can be flexibly adjusted. With the increase in the length of a plaintext m , the ratio between the length of a ciphertext and a plaintext, $\alpha = \frac{|m|+|p|+320}{|m|}$, becomes even closer to 1. Table 2 shows a comparison of the modified Zheng-Seberry_{uh} schemes with a few of the relevant encryption schemes.

Table 2. Efficiency comparison of the modified Zheng-Seberry_{uh} schemes with some relevant encryption schemes. “trapdoor permutation⁺” denotes trapdoor permutations that are uninvertible with access to a H -inverting oracle. “one-way hash⁺” denotes adaptively secure perfectly one-way hash. “SPD-OW” denotes set partial domain one-wayness. “SKE” denotes secure symmetric encryption. “MAC” denotes secure message authentication code. “Enc Exp” (“Dec Exp”) denotes the number of exponentiations or double exponentiations in encryption (decryption).

	Enc Exp	Dec Exp	Assumption	RO
Modified Zheng-Seberry _{uh}	2	1	adaptive DDH	No
Cramer-Shoup [10]	4	3	DDH	No
Kurosawa-Desmedt [22]	3	1	DDH, SKE	No
Hofheinz-Kiltz [17]	3	1	DDH	No
Hofheinz-Kiltz [18]	roughly 2	roughly 1	Rabin’s trapdoor OWP	No
DHIES [11]	2	1	ODH, SKE, MAC	No
Pandey-Pass-Vaikuntanathan [25]	-	-	trapdoor permutation ⁺ , one-way hash ⁺	No
Zheng-Seberry _{1wh} [2]	2	1	GDH	Yes
OAEP [15]	1	1	SPD-OW	Yes
Bellare-Rogaway [3]	-	-	trapdoor OWP	Yes

7 Concluding Remarks

We have proved the adaptive chosen ciphertext security of a modified version of Zheng and Seberry’s encryption scheme that employs universal hashing. The scheme investigated in this work is based on discrete logarithms in a subgroup. A possible interesting area for further research is to investigate whether similar results can be obtained with schemes built on other computationally hard problems, such as the integer factorization problem.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)

2. Baek, J., Zheng, Y.: Zheng and Seberry's public key encryption scheme revisited. *International Journal of Information Security (IJIS)* 2(1), 37–44 (2003)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: *First ACM Conference on Computer and Communication Security*, pp. 62–73. Association for Computing Machinery (1993)
4. Bellare, M., Rogaway, P.: Optimal asymmetric encryption—how to encrypt with RSA. In: De Santis, A. (ed.) *EUROCRYPT 1994*. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
5. Bellare, M., Rogaway, P.: Minimizing the use of random oracles in authenticated encryption schemes. In: Han, Y., Quing, S. (eds.) *ICICS 1997*. LNCS, vol. 1334, pp. 1–16. Springer, Heidelberg (1997)
6. Bernstein, D.J.: Polynomial evaluation and message authentication (2007), <http://cr.yp.to/papers.html#pema>
7. Boldyreva, A., Fischlin, M.: On the security of OAEP. In: Lai, X., Chen, K. (eds.) *ASIACRYPT 2006*. LNCS, vol. 4284, pp. 210–225. Springer, Heidelberg (2006)
8. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: *STOC 1998*, pp. 209–218. ACM Press, New York (1998)
9. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. *Journal of Computer and System Sciences* 18(2), 143–154 (1979)
10. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) *CRYPTO 1998*. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
11. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) *EUROCRYPT 2002*. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
12. den Boer, B.: A simple and key-economical unconditional authentication scheme. *Journal of Computer Security* 2(1), 65–71 (1993)
13. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. *SIAM Journal on Computing* 30(2), 391–437 (2000)
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) *CRYPTO 1999*. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
15. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. In: Kilian, J. (ed.) *CRYPTO 2001*. LNCS, vol. 2139, pp. 260–274. Springer, Heidelberg (2001)
16. Gennaro, R., Shoup, V.: A note on an encryption scheme of Kurosawa and Desmedt (2005), <http://www.shoup.net/papers/kdnote.pdf>
17. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) *CRYPTO 2007*. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
18. Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) *EUROCRYPT 2009*. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
19. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions. In: *STOC 1989*, pp. 12–24. ACM Press, New York (1989)
20. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. In: Joux, A. (ed.) *EUROCRYPT 2009*. LNCS, vol. 5479, pp. 589–608. Springer, Heidelberg (2009)
21. Krawczyk, H.: On extract-then-expand key derivation functions and an HMAC-based KDF (2008), <http://www.ee.technion.ac.il/~hugo/kdf/>

22. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
23. Leurent, G., Nguyen, P.Q.: How risky is the random oracle model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 445–464. Springer, Heidelberg (2009)
24. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen cipher-text attacks. In: STOC 1990, pp. 14–16. ACM Press, New York (1990)
25. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 57–74. Springer, Heidelberg (2008)
26. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
27. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)
28. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 239–259. Springer, Heidelberg (2001)
29. Soldera, D., Seberry, J., Qu, C.: The analysis of Zheng-Seberry scheme. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 159–168. Springer, Heidelberg (2002)
30. Zheng, Y., Seberry, J.: Immunizing public key cryptosystems against chosen ciphertext attacks. IEEE journal on selected areas in communications 11(5), 715–724 (1993)

Appendix

Definition 3. (*Family of adaptive one-to-one one-way functions*) [25]. A family of injective one-way functions $\mathcal{F} = \{f_{tag} : D_{tag} \rightarrow \{0, 1\}^*\}_{tag \in \{0, 1\}^n}$ is called *adaptively secure* if

- There is an efficient randomized domain sampler D , which on input $tag \in \{0, 1\}^n$, outputs a random element in D_{tag} . There is a deterministic polynomial algorithm M such that for all $tag \in \{0, 1\}^n$ and for all $x \in D_{tag}$, $M(tag, x) = f_{tag}(x)$.
- Let $\mathcal{O}_{tag}(\cdot, \cdot)$ denote an oracle that, on input tag' and y , outputs $f_{tag'}^{-1}(y)$ if $tag' \neq tag$, $|tag'| = |tag|$ and \perp otherwise. The family \mathcal{F} is *adaptively secure* if, for any probabilistic polynomial time adversary A which has access to the oracle $\mathcal{O}_{tag}(\cdot, \cdot)$, there exists a negligible function μ such that for all n , and for all tags $tag \in \{0, 1\}^n$,

$$\Pr[x \leftarrow D_{tag} : A^{\mathcal{O}_{tag}(\cdot, \cdot)}(tag, f_{tag}(x)) = x] \leq \mu(n)$$

where the probability is over the random choice of x and the coin tosses of A .

Definition 4. (*Adaptive PRG*) [25]. Let a family of functions $\mathcal{G} = \{G_{tag} : \{0, 1\}^n \rightarrow \{0, 1\}^{s(n)}\}_{tag \in \{0, 1\}^n}$ be a pseudorandom generator (PRG). And let

$\mathcal{O}_{tag}(\cdot, \cdot)$ denote an oracle that, on input (tag', y) such that $tag' \neq tag$, $|tag'| = |tag|$, outputs 1 if y is in the range of $G_{tag'}$ and 0 otherwise.

We say that \mathcal{G} is an adaptively secure PRG if, for any probability polynomial-time adversary A which has access to the oracle $\mathcal{O}_{tag}(\cdot, \cdot)$, there exists a negligible function μ such that for all n and for all tags $tag \in \{0, 1\}^n$,

$$|\Pr[y \leftarrow G_{tag}(U_n) : A^{\mathcal{O}_{tag}(\cdot, \cdot)}(y) = 1] - \Pr[y \leftarrow U_m : A^{\mathcal{O}_{tag}(\cdot, \cdot)}(y) = 1]| \leq \mu(n)$$

where the probability is over the random choice of y and the coin-tosses of the adversary A .

Proof of Claim [1](#), Claim [2](#) and Claim [3](#)

To show that Claim [1](#) holds, we first note that if $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+Q+W]}\}$ is an adaptive DDH quadruple and the event *Bad* does not happen, then the experiment perfectly simulates Game 1 and the adversary's views in the experiment and Game 1 are identical. Hence, we have

$$\Pr[\text{Game 1} \wedge \neg \text{Bad}] = \Pr[\text{Exp} \wedge \neg \text{Bad}]$$

Applying Lemma [1](#), we have $|\Pr[\text{Game 1}] - \Pr[\text{Exp}]| \leq \Pr[\text{Bad}]$, where $\Pr[\text{Bad}]$ is negligible. On the other hand, since $|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \geq 1/p(k)$, we have

$$\begin{aligned} & |\Pr[\text{Game 1}] - \Pr[\text{Exp}]| + |\Pr[\text{Game 2}] - \Pr[\text{Exp}]| \\ & \geq |\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \\ & \geq 1/p(k). \end{aligned}$$

Therefore, $|\Pr[\text{Game 2}] - \Pr[\text{Exp}]|$ is non-negligible, from which Claim [1](#) follows.

Using a similar argument to the correctness of Claim [1](#), we have Claim [2](#). Summing up Claim [1](#) and Claim [2](#), the adaptive DDH assumption can be compromised by observing the behavior of the adversary. Specifically, if $|\Pr[\text{Game 1}] - \Pr[\text{Exp}]|$ is negligible, then $|\Pr[\text{Game 2}] - \Pr[\text{Exp}]|$ must be non-negligible. In this case, $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+Q+W]}\}$ must be an adaptive DDH quadruple. Likewise, if $|\Pr[\text{Game 1}] - \Pr[\text{Exp}]|$ is non-negligible, then $|\Pr[\text{Game 2}] - \Pr[\text{Exp}]|$ must be negligible. In this case, $\{g, g^a, g^b, G_{g^a, g^b}(g^c)_{[1, \dots, P+Q+W]}\}$ must not be an adaptive DDH quadruple. These lead to Claim [3](#).

A Public-Key Traitor Tracing Scheme with an Optimal Transmission Rate*

Yi-Ruei Chen and Wen-Guey Tzeng

Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan 30010, ROC
yrchen.cs98g@nctu.edu.tw, wgtzeng@cs.nctu.edu.tw

Abstract. The way of transmitting the encrypted digital content to the legitimate subscribers over a broadcast channel has wide commercial applications, such as Pay-TV, DVD, etc. In order to discourage the legitimate subscribers from giving away their decryption keys, the traitor tracing scheme comes up. In this paper, we propose a public-key traitor tracing scheme that has optimal transmission rate. In other words, our scheme enables everyone to transmit the encrypted digital contents almost without any redundancy. As for tracing, our scheme supports black-box tracing, i.e., identifying colluders without opening the pirate decoder. Moreover, in our scheme, the storage requirement for legitimate subscribers and digital content broadcasters is smaller than that of previous schemes.

Keywords: Traitor tracing, transmission rate, fingerprinting code.

1 Introduction

Consider the scenario that a data supplier distributes the digital content over a broadcast channel. The data supplier gives a secret key to each legitimate subscriber. Then the data supplier broadcasts the encrypted digital content and the legitimate subscribers decrypt the digital content by their secret keys. The protection for some Pay-TV, CD-ROM, DVD, and online databases is based on this scenario. However, some malicious subscribers (called *traitors*) might give copies of their secret keys to illegitimate users (called *pirates*). Then the pirates decrypt the digital content for free. In order to solve the problem above, the *traitor tracing* scheme comes up.

The goal of traitor tracing schemes is to discourage legitimate subscribers from giving away their secret keys. One approach is to give each subscriber a unique set of secret keys that both decrypt the encrypted digital content and identify (“trace”) the subscribers. To avoid being traced by a tracer, the traitors may collude to obfuscate their secret keys and generate a new secret key set (called

* The research was supported in part by projects NSC 96-2628-E-009-011-MY3 and 98-2219-E-009-003-

pirate key). We call a traitor tracing scheme *t-collusion resistant* if at least one of the traitors can be identified when t traitors collude to generate a pirate key in this way. If t is the number of all legitimate users, the traitor tracing scheme is called *fully-collusion resistant*. Note that the traitors may embed the pirate keys into a “tamper-resistant” hardware (called *pirate decoder*) to prevent the tracer from reading any data inside. So, during the tracing, the tracer has to treat the pirate decoder as a *black box* – only the outcome of a pirate decoder can be examined.

In many previous traitor tracing schemes, the overhead of broadcasting the encrypted digital content is proportional to the number of legitimate subscribers. But in some applications, such as Pay-TV, the number of legitimate subscribers might be up to millions. This is a great burden. The approach of *public-key traitor tracing* schemes is to enable everyone (e.g. Pay-TV stations) to broadcast the encrypted digital content. The *public traceability* of a traitor tracing scheme allows everyone with a pirate decoder to trace the traitors. In order to measure the efficiency of traitor tracing schemes, we consider the “transmission rate” of encrypted digital content (“ciphertext”), that is, the ratio of the size of ciphertext to the size of the digital content. We also care about the storage requirements of subscribers’ secret keys, and the broadcast keys.

Related work. The *traitor tracing* scheme was first introduced by Chor, Fiat and Naor [9], and later refined in [15,10]. The concept of *public-key traitor tracing* schemes was proposed in Kurosawa and Desmedt [14], and Boneh and Franklin [2]. The traitor tracing schemes in [2,3,8,12,11,14,13,16,18,21,22] are public-key traitor tracing schemes. In [8], Chabanne, Phan, and Pointcheval proposed the concept of *public traceability*. A class of traitor tracing schemes relying on the usage of *fingerprinting codes* [5,20] was introduced by Kiayias and Yung [13]. They showed that if the plaintexts are large (e.g. multimedia content), it is possible to obtain constant transmission rate. For example, the schemes in [8,18,17,11] have constant transmission rate. While considering the transmission rate, we have two main categories in the traitor tracing schemes:

- Schemes with *no constant transmission rate* [2,4]: These schemes are well-suited to encrypt small digital content (usually using for the session-key exchanges in the “hybrid encryption”). The user-key size and the public-key size are often relatively small in these schemes. But the transmission rate in these schemes is often linear or sublinear to the maximal number of colluders.
- Schemes with *constant transmission rate* [13,8,11] (including ours): These schemes are well-suited to encrypt large digital content (e.g. multimedia content). They are all constructed by using the fingerprinting codes. One advantage of these schemes is that they often have efficient black-box tracing algorithms. Nevertheless, the user-key size and the public-key size are often relatively large (according to the codeword length in the fingerprinting codes).

Our Contributions. We propose a public-key traitor tracing schemes with efficient black-box tracing and the optimal transmission rate. The storage

Table 1. Scheme Comparison

	transmission rate	user-key size	public-key size	black-box tracing	traceability
BF99 [2]	$2t + 1$	$2t$	$2t + 1$	inefficient	private
BSW06 [4]	$6\sqrt{N}$	1	$4\sqrt{N} + 2$	O	public
KY02 [13]	~ 3	2ℓ	4ℓ	O	private
CPP05 [8]	~ 1	2ℓ	$\ell + 1$	X	private
FNP07 [11]	~ 1	2ℓ	10ℓ	O	private
Ours	~ 1	$\ell + 2$	$2\ell + 1$	O	private

[†] ℓ : the codeword length in fingerprinting code

[†] N : the total number of legitimate subscribers

requirements in our scheme for user-keys and public-keys are smaller than previous schemes that have the constant (or optimal) transmission rate. Our scheme is based on a fingerprinting code, and an all-or-nothing transformation [6,7,19]. The idea is to encrypt a block of the output of an all-or-nothing transformation by a special public-key scheme. The encryption does not entail much overhead and allows us to feed indistinguishable messages for tracing. The comparison with other related schemes is given in Table 1. We show that our scheme is semantically secure based on the DDH assumption and the indistinguishability of PKE-AONT. We also show that our traitor tracing scheme is t -collusion resistant under the DDH assumption.

2 Preliminaries

Notations. A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* if for every constant $c \in \mathbb{N}$, there exists an integer $k_0 \in \mathbb{N}$ such that $f(k) \leq k^{-c}$ for all $k \geq k_0$, denoted by $neg(k)$.

We use $x \stackrel{\$}{\leftarrow} X$ to denote that x is chosen from the set X uniformly. Let \mathcal{M} be the plaintext space.

Fingerprinting Codes. The fingerprinting technique with fingerprinting codes embeds a specific fingerprint (*codeword*) to each document copy so that one can identify which copy of document by examining the embedded fingerprint. The codeword is a collection of some alphabets. The traitors will collude and try to modify their codewords to avoid being identified. However, the coalition of the traitors is restricted by the *marking assumption*: the traitors are only able to compare their codewords and make a modification from their respective codewords differing in some positions. Under the marking assumption, the possible modified codeword set from t traitor’s codewords set W is called a *feasible set* of W .

- For a codeword $w \in \{0, 1\}^\ell$, we write $w = w_1w_2\dots w_\ell$, where $w_i \in \{0, 1\}$.
- Let $W = \{w^{(1)}, \dots, w^{(t)}\} \subseteq \{0, 1\}^\ell$. We say that a codeword \bar{w} is *feasible* for W if $\forall i \in \{1, 2, \dots, \ell\} \exists j \in \{1, 2, \dots, t\}$ s.t. $\bar{w}_i = w_i^{(j)}$. For example, if $W = \{0101, 1111\}$, the codewords $\{0101, 0111, 1101, 1111\}$ are feasible for W .
- For a codeword set $W \subseteq \{0, 1\}^\ell$, we say that the feasible set of W , denoted by $F(W)$, is the set of all codewords that are feasible for W .

Table 2. Length of the fingerprinting codes with respect to the number N of codewords

	t -collision resistant	fully-collision resistant
BS98 [5]	$\ell = O(t^4 \log(N/\epsilon) \log(1/\epsilon))$	$\ell = O(N^3 \log(N/\epsilon))$
T03 [20]	$\ell = O(t^2 \log(N/\epsilon))$	$\ell = O(N^2 \log(N/\epsilon))$

A fingerprinting code scheme consists of two probabilistic polynomial-time algorithms: the *codeword generation* algorithm G and the *codeword tracing* algorithm T . Algorithm G generates codeword set $\Gamma = \{w^{(1)}, \dots, w^{(N)}\} \in (\{0, 1\}^\ell)^N$ for some $\ell > 0$ and the trace-key tk . By taking a pirate codeword \bar{w} and tk as input, algorithm T outputs at least one of the traitors who collude to generate \bar{w} . The fingerprinting codes enable a data supplier to distribute an “object” with many fingerprinting copies. Assume $\Gamma \in (\{0, 1\}^\ell)^N$ is a codeword set generated by the algorithm G , and denote a L -length object to be distributed by P . First we partition the object P into ℓ blocks and embed exactly one “mark” in each block. Let $P_{j,s}$ be the j -th block which contains the j -th mark with state $s \in \{0, 1\}$. For each block, choose a random key $K_{j,s}$, the distributed data will be $\{f_{K_{j,s}}(P_{j,s}) | 1 \leq j \leq \ell, s \in \{0, 1\}\}$, where f is an encryption scheme. Let $w^{(u)} \in \Gamma$ be the codeword in Γ and associated with user u . The private user-key for u is $\{K_{1,w_1^{(u)}}, K_{2,w_2^{(u)}}, \dots, K_{\ell,w_\ell^{(u)}}\}$, and $P(w^{(u)}) = P_{1,w_1^{(u)}} || P_{2,w_2^{(u)}} || \dots || P_{\ell,w_\ell^{(u)}}$ will be the copy of P implied by $w^{(u)}$.

Boneh and Shaw [5] constructed a fully-collision resistant fingerprinting code as well as t -collision resistant secure codes. Tardos [20] proposed a shorter code. The comparison of their codeword lengths is in Table 2, where N is the number of codewords, and ϵ is the security parameter.

All-Or-Nothing Transformation. An all-or-nothing transformation (AONT) Σ is an efficient, unkeyed, and randomized transformation with the property that it is hard to invert unless the entire output is known [19]. Σ maps an ℓ' -block sequence x , together with a random string ρ to an ℓ -block sequence y with the following properties:

- Given x and ρ , $y \stackrel{\$}{\leftarrow} \Sigma(x; \rho)$ can be computed efficiently.
- Given all blocks of y , $x \leftarrow \Sigma^{-1}(y)$ can be computed efficiently.
- It is infeasible to get any information about x if any block of y is missing.

Notice that the AONT expands plaintext size by roughly $1 + 1/\ell$. This results in an asymptotical unitary ciphertext-to-plaintext ratio.

Decisional Diffie-Hellman (DDH) Assumption. For a cyclic group \mathbb{G} with a generator g . Let \mathcal{V} be the distribution $\{(g, g^u, g^v, g^{uv})\}$ and \mathcal{R} be the distribution $\{(g, g^u, g^v, g^w)\}$. For any polynomial time adversary \mathcal{A} , \mathcal{A} distinguishes the two distributions \mathcal{V} and \mathcal{R} with negligible function of λ , i.e., $|\Pr[\mathcal{A}(X) = 1 : X \in \mathcal{V}] - \Pr[\mathcal{A}(X) = 1 : X \in \mathcal{R}]| = \text{neg}(|\mathbb{G}|)$.

3 The Notion for Public-Key Traitor Tracing Scheme

A public-key traitor tracing scheme is a 4-tuple of probabilistic polynomial-time algorithms (Setup, Encrypt, Decrypt, Trace), where

Setup($1^\lambda, N$). The setup takes as input a security parameter λ and N , the number of users in the system. The algorithm outputs a public broadcast-key BK, a secret trace-key TK, and the private user-key SK_u for each legitimate subscriber u .

Encrypt(BK, M). The encryption algorithm takes as input the public broadcast-key BK and a message $M \in \mathcal{M}$. The algorithm outputs a ciphertext C .

Decrypt(SK_u, C). The decryption algorithm takes as input the private user-key SK_u of user u and a ciphertext C . The algorithm outputs a message M or \perp .

Trace $^{\mathcal{D}}$ (TK). The tracing algorithm takes as input the private trace-key TK and queries the pirate decoder \mathcal{D} as a black-box oracle. The algorithm outputs a traitor set S which is a subset of $\{1, \dots, N\}$.

Moreover, the scheme must satisfy the correctness property as follows:

For all $u \in \{1, \dots, N\}$ and for all $M \in \mathcal{M}$: if $(BK, TK, (SK_1, \dots, SK_N)) \stackrel{\$}{\leftarrow}$ Setup($1^\lambda, N$) and $C \stackrel{\$}{\leftarrow}$ Encrypt(BK, M), then Decrypt(SK_u, C) = M .

Semantic Security Game

- *Setup*. The challenger runs Setup, and gives BK to the adversary.
- *Challenge*. The adversary chooses two plaintexts $M_0, M_1 \in \mathcal{M}$ to the challenger. Then the challenger flips a coin $b \in \{0, 1\}$, and gives a ciphertext $C_b \stackrel{\$}{\leftarrow}$ Encrypt(BK, M_b) to the adversary.
- *Guess*. The adversary returns a guess $b' \in \{0, 1\}$ of b to the challenger.

The advantage of winning this game by the adversary is $\text{Adv}_{SS}^{\text{TTS}} := |\Pr[b' = b] - \frac{1}{2}|$

Definition 1 (Semantically secure). An N -user public-key traitor tracing scheme is semantically secure if for all polynomial time adversaries \mathcal{A} , $\text{Adv}_{SS}^{\text{TTS}}$ is a negligible function of the security parameter.

Traceable against t -collusion Game

- *Setup*. The challenger runs Setup and gives BK to the adversary. The adversary chooses a traitor set $T = \{u_1, \dots, u_t\} \subseteq \{1, \dots, N\}$ to the challenger. Then the challenger gives the adversary $SK_{u_1}, \dots, SK_{u_t}$ to produce a pirate decoder \mathcal{D} .
- *Trace*. By taking a pirate decoder \mathcal{D} as a decryption oracle, the challenger runs the algorithm Trace $^{\mathcal{D}}$ (TK) to obtain a traitor set $S \subseteq \{1, \dots, N\}$.

The adversary wins this game if (1) \mathcal{D} decrypts all valid ciphertext with a constant probability δ , i.e., $\Pr[\mathcal{D}(\text{Encrypt}(\text{BK}, M)) = M] \geq \delta$, and (2) $S \cap T \neq \emptyset$.

The probability of adversary winning this game is $\text{Adv}_{TR}^{\text{TTS}}$.

Definition 2 (Traceable against t -collusion). An N -user public-key traitor tracing scheme is traceable against t -collusion if for all polynomial time adversaries \mathcal{A} of corrupting t users and any constant $\delta > 0$, $\text{Adv}_{\text{TR}}^{\text{TTS}}$ is a negligible function of the security parameter.

4 A Public-Key Traitor Tracing Scheme for Two Users

In this section, we give a construction of a public-key traitor tracing scheme for two users. Then we show that this scheme is semantically secure and traceable against 1-collusion. Indeed, this scheme will be a subscheme in the construction of our public-key traitor tracing scheme for N users in section 5.

4.1 Our Construction

Our public-key traitor tracing scheme for two users is 2-PK-TTS = (2-Setup, 2-Encrypt, 2-Decrypt, 2-Trace), where

2-Setup(1^λ) Given a security parameter λ , the algorithm generates a λ -bit prime q , a cyclic group \mathbb{G} of order q , and a generator g of \mathbb{G} . Then the algorithm chooses $f(x) = a_0 + a_1x \pmod{q}$, where $a_0, a_1 \xleftarrow{\$} \mathbb{Z}_q^*$ and sets

- Public broadcast-key $bk := \langle g, (g^{a_0}, g^{a_1}) \rangle$
- Secret trace-key $tk := \langle f(x) \rangle$
- User-key $sk_\sigma := \langle i_\sigma, f(i_\sigma) \rangle$, where $i_\sigma \in \mathbb{Z}_q^*, \forall \sigma \in \{0, 1\}$

2-Encrypt(bk, m) Given bk and a plaintext $m \in \mathcal{M}$, the algorithm chooses $r, j \xleftarrow{\$} \mathbb{Z}_q^*$, where $j \neq i_0$ or i_1 , computes $g^{rf(j)} = (g^{a_0}(g^{a_1})^j)^r$ and outputs the ciphertext $c := \langle mg^{ra_0}, g^r, (j, g^{rf(j)}) \rangle$.

2-Decrypt(sk_σ, c) Given a ciphertext $c = \langle A, R, (j, W) \rangle$ and a user-key sk_σ , the algorithm computes the plaintext based on the lagrange interpolation:

$$m = A/W \frac{-i_\sigma}{j-i_\sigma} R^{f(i_\sigma) \frac{-j}{i_\sigma-j}}.$$

2-Trace ^{\mathcal{D}} (tk) Given a pirate decoder \mathcal{D} that decrypts all valid ciphertext perfectly as a decryption oracle. The algorithm does:

1. **2-TrEncrypt**(bk, m) The algorithm chooses $r, \hat{r}, j \xleftarrow{\$} \mathbb{Z}_q^*$ and computes a probe ciphertext $\hat{c} \xleftarrow{\$} \langle A = mg^{ra_0}, R = g^r, (j, \hat{W} = g^{\hat{r}f(j)}) \rangle$.
2. $\forall \sigma \in \{0, 1\}$, pre-compute $V_\sigma = \hat{W} \frac{-i_\sigma}{j-i_\sigma} R^{f(i_\sigma) \frac{-j}{i_\sigma-j}}$.
3. $\forall \sigma \in \{0, 1\}$, if $\mathcal{D}(\hat{c}) = A/V_\sigma$, output $S = \{\sigma\}$; else output $S = \{0, 1\}$.

4.2 Security Analysis of our 2-PK-TTS scheme

Theorem 1. *The 2-PK-TTS scheme is semantically secure under the DDH assumption.*

Proof. By contradiction, assume that there exists a 2-PK-TTS scheme adversary \mathcal{A} that wins the semantic security game with a non-negligible advantage $\epsilon > 0$. We construct an algorithm \mathcal{B} that breaks the DDH assumption with non-negligible advantage ϵ as follows:

- *Setup.* Algorithm \mathcal{B} is given as input an instance (g, g^u, g^v, X) of the DDH assumption, and it wants to determine whether $X = g^{uv}$ or X is a random element in \mathbb{G} (\mathbb{G} has prime order q). \mathcal{B} chooses $a_0, a_1 \xleftarrow{\$} \mathbb{Z}_q^*$ and sets $bk = \langle g, (g^u, g^{a_1}) \rangle$ to \mathcal{A} . (we see that $f(x) = u + a_1x \pmod{q}$)
- *Challenge.* \mathcal{A} chooses two plaintexts $m_0, m_1 \in \mathcal{M}$ to \mathcal{B} , then \mathcal{B} flips a coin $b \in \{0, 1\}$, and sets the challenge $c_b = \langle m_b X, g^v, (j, X(g^v)^{a_1 j}) \rangle$ to \mathcal{A} , where $j \xleftarrow{\$} \mathbb{Z}_q^*$.
- *Guess.* \mathcal{A} outputs $b' \in \{0, 1\}$ to \mathcal{B} . If $b' = b$, \mathcal{B} answers that $X = g^{uv}$; else \mathcal{B} answers that X is a random element in \mathbb{G} .

If $X = g^{uv}$, \mathcal{A} gets a valid ciphertext $c_b = \langle m_b g^{uv}, g^v, (j, g^{v(u+a_1 j)}) \rangle$. Therefore, \mathcal{A} answers $b' = b$ successfully with probability $\frac{1}{2} + \epsilon$;

If X is a random element in \mathbb{G} , \mathcal{A} gets an invalid ciphertext. In this case, \mathcal{A} answers $b' = b$ successfully with probability $\frac{1}{2}$.

Hence, \mathcal{B} solves the DDH problem with non-negligible advantage ϵ . This is a contradiction to the DDH assumption. So we conclude that such adversary \mathcal{A} does not exist.

Theorem 2. *The 2-PK-TTS scheme is traceable against 1-collusion under the DDH assumption.*

Proof. By contradiction, assume that there exists an adversary \mathcal{A} that, given the public-key bk and one of user-keys sk_σ in 2-PK-TTS scheme, \mathcal{A} produces a pirate decoder \mathcal{D} that decrypts all valid ciphertexts perfectly, i.e., $\Pr[\mathcal{D}(\text{2-Encrypt}(bk, m)) = m : \mathcal{D} \xleftarrow{\$} \mathcal{A}(bk, sk_\sigma), \sigma \in \{0, 1\}] = 1$. But when given a probe ciphertext \hat{c} , \mathcal{D} outputs a different value from the pre-computed values in 2-Trace algorithm with non-negligible probabilistic $\epsilon > 0$, i.e., $\Pr[\mathcal{D}(\hat{c}) \neq A/V_\sigma] = \epsilon$. We construct an algorithm \mathcal{B} that breaks the DDH assumption with non-negligible advantage $\frac{\epsilon}{2}$ as follows:

- *Setup.* Algorithm \mathcal{B} is given as input an instance (g, g^u, g^v, X) of DDH assumption, and it wants to determine whether $X = g^{uv}$ or X is a random element in \mathbb{G} . \mathcal{B} chooses $i, z \xleftarrow{\$} \mathbb{Z}_q^*$ and gives $\mathcal{A} bk = \langle g, (g^u, g^{a_1 = (\frac{g^z}{g^u})^{i-1}}) \rangle$. \mathcal{A} chooses a traitor set $\mathbb{T} = \{0\}$ or $\{1\}$ to \mathcal{B} . Then \mathcal{B} gives $\mathcal{A} sk = \langle i, z \rangle$ to produces a pirate decoder \mathcal{D} .
- *Trace.* By taking a pirate decoder \mathcal{D} as a decryption oracle, \mathcal{B} runs the modified 2-Trace $_{\mathbb{T}}$ as follows:
 1. Choose $A \xleftarrow{\$} \mathbb{G}$, and $j \xleftarrow{\$} \mathbb{Z}_q^*$, where $j \neq i$. Compute $W = X(\frac{(g^v)^z}{X})^{ji-1}$ and set the ciphertext as $\bar{c} \leftarrow \langle A, g^v, (j, W) \rangle$.
 2. Pre-compute $V \leftarrow W^{\frac{-i}{j-i}}(g^v)^z \frac{-j}{i-j}$.
 3. If $\mathcal{D}(\bar{c}) = A/V$, \mathcal{B} answers that $X = g^{uv}$ or X is a random element in \mathbb{G} randomly; else \mathcal{B} answers that X is a random element in \mathbb{G} .

If $X = g^{uv}$, ciphertext \bar{c} is a valid ciphertext, since

$$X(\frac{(g^v)^z}{X})^{ji-1} = g^{uv}(\frac{(g^v)^z}{g^{uv}})^{ji-1} = g^{uv}((\frac{g^z}{g^u})^{i-1})^{vj} = g^{uv}(g^{a_1})^{vj} = g^{v(u+a_1 j)}.$$

In this case, $\mathcal{D}(\bar{c}) = A/V$, \mathcal{B} gives the correct answer with probability $\frac{1}{2}$;

If X is a random element in \mathbb{G} , ciphertext \bar{c} is an invalid ciphertext. In this case, $\mathcal{D}(\bar{c}) \neq A/V$ with probability ϵ , and $\mathcal{D}(\bar{c}) = A/V$ with probability $1 - \epsilon$. Therefore, \mathcal{B} gives the correct answer with probability $\epsilon + \frac{1}{2}(1 - \epsilon) = \frac{1}{2} + \frac{\epsilon}{2}$.

Hence, \mathcal{B} solves the DDH problem with non-negligible advantage $\frac{\epsilon}{2}$. This is a contradiction to the DDH assumption. So we conclude that such adversary \mathcal{A} does not exist.

5 A Public-Key Traitor Tracing Scheme for N Users

Our construction is based on the use of our 2-PK-TTS scheme, an AONT Σ and a fingerprinting code $\Gamma = \{w^{(1)}, \dots, w^{(N)}\}$ over $\{0, 1\}^\ell$. At a high level, the idea is to “concatenate” ℓ instance of 2-PK-TTS scheme according to the code Γ . Each legitimate subscriber $u \in \{1, \dots, N\}$ is associated to a codeword $w^{(u)}$ in Γ and assigned a private user-key set $\text{SK}_u = \{sk_{1, w_1^{(u)}}, \dots, sk_{\ell, w_\ell^{(u)}}\}$, where $w_j^{(u)}$ is the j -th bit of the codeword $w^{(u)}$, and $sk_{j,0}, sk_{j,1}$ are the keys for the j -th instance of the 2-PK-TTS scheme. For example, let $\ell = 3$. If the codeword corresponding to legitimate subscriber u is 011, its user-key set is $\{sk_{1,0}, sk_{2,1}, sk_{3,1}\}$. For tracing, we use the j -th 2-PK-TTS to identify the j -th symbol of the pirate codeword, for all $j \in \{1, 2, \dots, \ell\}$. Finally, by the tracing algorithm in the fingerprinting code, we find the collusion codeword set for constructing a pirate codeword, i.e., we find the collusion traitor set.

In order to achieve the optimal transmission rate, we notice the cryptosystem PKE-AONT proposed by Zhang, Hanaoka, and Imai [23]. It encrypts some bits of the output of an AONT by a public-key encryption scheme. Given an AONT Σ and a public-key encryption scheme (G, E, D) , PKE-AONT first transforms the original message M' into an all-or-nothing message $M = m_1 || \dots || m_\ell$ by Σ and then randomly chooses a block of M to encrypt it as $C = m_1 || \dots || m_{k-1} || E(pk, m_k) || m_{k+1} || \dots || m_\ell$. For decrypting the ciphertext C , the decryption algorithm first decrypts the k -th block to recover M and compute the original message M' by Σ .

5.1 Our Construction

For convenience, we introduce some notations in our scheme:

- Let Σ be an AONT that maps an ℓ' -block sequence together with a random string to an ℓ -block sequence.
- $\text{MINUS}_k(M)$ Given an $\ell\lambda$ -bit message $M = m_1 || \dots || m_\ell$ and a position index $k \in \{1, \dots, \ell\}$, the algorithm “minus” the k -th block of M , i.e., $\text{MINUS}_k(M) = m_1 || \dots || m_{k-1} || m_{k+1} || \dots || m_\ell$.
- $\text{COMB}_k(Y, m)$ Given an $(\ell - 1)\lambda$ -bit message $Y = y_1 || \dots || y_{\ell-1}$, a λ -bit message m and a position index $k \in \{1, \dots, \ell - 1\}$, the algorithm first splits Y into $X_1 || X_2$, where X_1 is the front $(k - 1)\lambda$ bits of Y and X_2 is the rest bits of Y . The algorithm “combines” and outputs the messages with order X_1, m , and X_2 , i.e. $\text{COMB}_k(Y, m) = y_1 || \dots || y_{k-1} || m || y_k || \dots || y_{\ell-1}$.

Our traitor tracing scheme for N users $\Pi = (\text{Setup}, \text{Encrypt}, \text{Decrypt}, \text{Trace})$ is as follows:

Setup($1^\lambda, N$). Given a security parameter λ and user number N , the algorithm generates a fingerprinting code $\Gamma = \{w^{(1)}, \dots, w^{(N)}\} \in (\{0, 1\}^\ell)^N$ for some ℓ . Then it runs **2-Setup** ℓ times to generate the keys $\langle (bk_i, tk_i, (sk_{0,i}, sk_{1,i}))_{i=1}^N \rangle$ (but use the same $q, \mathbb{G}, g, i_0, i_1$) and sets

- Public broadcast-key $\text{BK} := \langle g, (g^{a_{0,j}}, g^{a_{1,j}})_{j=1}^\ell \rangle$
 (we denote the k -th key of BK by $\text{BK}_k = (g, (g^{a_{0,k}}, g^{a_{1,k}}))$)
- Secret trace-key $\text{TK} := \langle (f_j(x))_{j=1}^\ell \rangle$
- User-key $\text{SK}_u := \langle w^{(u)}, i_0, i_1, (f_j(i_{w_j^{(u)}}))_{j=1}^\ell \rangle, \forall u \in \{1, 2, \dots, N\}$
 (we denote k -th key of SK_u by $\text{SK}_{u,k} = (i_{w_k^{(u)}}, f_k(i_{w_k^{(u)}}))$)

Encrypt(BK, M'). Given BK and a plaintext $M' \in \mathcal{M}^{\ell'}$, the algorithm chooses a random string $\rho \xleftarrow{\$} \{0, 1\}^\tau$, and computes $\Sigma(M'; \rho) = M = m_1 || \dots || m_\ell$. Then it chooses a position index $k \xleftarrow{\$} \{1, 2, \dots, \ell\}$, and computes the ciphertext $C \xleftarrow{\$} \langle k, \text{2-Encrypt}(\text{BK}_k, m_k), \text{MINUS}_k(M) \rangle$.

Decrypt(SK_u, C). Given a ciphertext $C = \langle k, c_k, Y \rangle$, user u computes $m_k \leftarrow \text{2-Decrypt}(\text{SK}_{u,k}, c_k)$ and $M' = \Sigma^{-1}(\text{COMB}_k(Y, m_k))$.

Trace ^{\mathcal{D}} (TK). Given a pirate decoder \mathcal{D} that decrypts all valid ciphertext perfectly as a decryption oracle. The algorithm does:

- For each position index $k \in \{1, 2, \dots, \ell\}$,
 1. Compute $\Sigma(M'; \rho) = M = m_1 || m_2 || \dots || m_\ell$, where $M' \xleftarrow{\$} \mathcal{M}^{\ell'}$ and $\rho \xleftarrow{\$} \{0, 1\}^\tau$.
 2. Call $\text{2-TrEncrypt}(\text{BK}_k, m_k) \xrightarrow{\$} \hat{c}_k = \langle A_k = m_k g^{r a_{0,k}}, R = g^r, (j, \hat{W}_k = g^{\hat{r} f_k(j)}) \rangle$. Set the probe ciphertext as $\hat{C} \xleftarrow{\$} \langle k, \hat{c}_k, Y = \text{MINUS}_k(M) \rangle$.
 3. $\forall \sigma \in \{0, 1\}$, pre-compute $M_{k,\sigma} = \text{COMB}_k(Y, A_k / \hat{W}_k^{\frac{-i_\sigma}{j-i_\sigma}} R^{f_k(i_\sigma) \frac{-j}{i_\sigma-j}})$.
 4. $\forall \sigma \in \{0, 1\}$, if $\Sigma(\mathcal{D}(\hat{C}); \rho) = M_{k,\sigma}$, set $w_k^* = \sigma$; else set $w_k^* = 0$ for convenience.
- Recover $w^* = w_1^* w_2^* \dots w_\ell^*$, then call the tracing algorithm in fingerprinting code by taking w^* as the input to obtain collude codewords. Finally, output the corresponding traitor set S .

5.2 Security Analysis of Our Scheme

Theorem 3. *The scheme Π is semantically secure under the semantic security of 2-PK-TTS and the indistinguishability of PKE-AONT.*

Proof. For each position index $k \in \{1, 2, \dots, \ell\}$, we use two games to bound the advantage of semantically secure in Π with $\text{Adv}_{\text{SS}}^{2\text{-PK-TTS}}$ and $\text{Adv}_{\text{ind}}^{\text{PKE-AONT}}$ as follows:

Game \mathbf{G}_0 . Define \mathbf{G}_0 as the original semantic security game and let S_0 be the event where $b' = b$, i.e., $\text{Adv}_{\text{SS}}^\Pi := |\Pr[S_0] - \frac{1}{2}|$.

Game \mathbf{G}_1 . This game is identical to \mathbf{G}_0 except that in the *Encrypt*, rather than being set $A_k = m_k g^{r^{a_0,k}}$, in \mathbf{G}_1 A_k is a random λ -bit string, i.e., $C \stackrel{\$}{\leftarrow} \langle A_k \stackrel{\$}{\leftarrow} \{0,1\}^\lambda, R = g^r, (j, W_k = g^{r^{f_k(j)}}) \rangle$, and we let S_1 be the event that $b' = b$ in this game.

Claim: $|\Pr[S_0] - \Pr[S_1]| \leq 2\text{Adv}_{\text{SS}}^{2\text{-PK-TTS}}$.

By reduction, if there exists an adversary \mathcal{A} that distinguishes the challenge of \mathbf{G}_0 and \mathbf{G}_1 with non-negligible probability $\epsilon > 0$, we use \mathcal{A} to construct an adversary \mathcal{B} for breaking the semantic security of 2-PK-TTS scheme with non-negligible advantage as follows:

- *Setup.* Algorithm \mathcal{B} is given as input an instance $bk = \langle g, (g^{a_0}, g^{a_1}) \rangle$ of 2-PK-TTS scheme and wants to determine whether the challenge C is construct by \mathbf{G}_0 or \mathbf{G}_1 . \mathcal{B} chooses $a_{0,j}, a_{1,j} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$, $\forall j \in \{1, 2, \dots, \ell\} \setminus \{k\}$, lets $g^{a_0,k} = g^{a_0}, g^{a_1,k} = g^{a_1}$, and sets $\text{BK} = \langle g, (g^{a_{0,j}}, g^{a_{1,j}})_{j=1}^\ell \rangle$ to \mathcal{A} .
- *Challenge.* \mathcal{A} chooses two plaintexts $M_0, M_1 \in \mathcal{M}^{\ell'}$ to \mathcal{B} , then \mathcal{B} flips a coin $b' \in \{0,1\}$, and it calls $\Sigma(M_{b'}, \rho) = m_{b',1} || m_{b',2} || \dots || m_{b',\ell}$, lets $m_{1-b',k} \stackrel{\$}{\leftarrow} \{0,1\}^\lambda$, and sends $m_{b'} = m_{b',k}, m_{1-b'} = m_{1-b',k}$ to 2-PK-TTS scheme challenger. Then 2-PK-TTS scheme challenger flips a coin $b \in \{0,1\}$ and sets the challenge $c_b \stackrel{\$}{\leftarrow} 2\text{-Encrypt}(bk, m_b)$ to \mathcal{B} . Finally, \mathcal{B} sends \mathcal{A} the challenge $C_{b'} = \langle k, c_b, Y = m_{b',1} || \dots || m_{b',k-1} || m_{b',k+1} || \dots || m_{b',\ell} \rangle$.
- *Guess.* \mathcal{A} outputs $\hat{b} \in \{0,1\}$ to \mathcal{B} . Then \mathcal{B} gives \hat{b} as its guess to 2-PK-TTS scheme challenger.

By the above construction, we see that \mathcal{B} “interpolates” between \mathbf{G}_0 and \mathbf{G}_1 for \mathcal{A} :

- If $b' = b$, \mathcal{A} gets a challenge in \mathbf{G}_0 ;
- If $b' = 1 - b$, \mathcal{A} gets a challenge in \mathbf{G}_1 .

Thus, it holds that $\Pr[S_0] = \Pr[\hat{b} = b' | b' = b]$ and $\Pr[S_1] = \Pr[\hat{b} = b' | b' = 1 - b]$, and we get

$$\begin{aligned} \Pr[\hat{b} = b] &= \Pr[\hat{b} = b | b' = b] \Pr[b' = b] + \Pr[\hat{b} = b | b' = 1 - b] \Pr[b' = 1 - b] \\ &= \frac{1}{2} (\Pr[\hat{b} = b | b' = b] + \Pr[\hat{b} = b | b' = 1 - b]) \\ &= \frac{1}{2} (\Pr[\hat{b} = b | b' = b] + 1 - \Pr[\hat{b} = 1 - b | b' = 1 - b]) \\ &= \frac{1}{2} + \frac{1}{2} (\Pr[\hat{b} = b' | b' = b] - \Pr[\hat{b} = b' | b' = 1 - b]) \\ &= \frac{1}{2} + \frac{1}{2} (\Pr[S_0] - \Pr[S_1]). \end{aligned}$$

It follows that $|\Pr[S_0] - \Pr[S_1]| = 2|\Pr[\hat{b} = b] - \frac{1}{2}| = 2\text{Adv}_{\text{SS}}^{2\text{-PK-TTS}}$.

To conclude the proof, due to indistinguishability of PKE-AONT, the adversary distinguishes the ciphertexts in \mathbf{G}_1 and the random bit string (as the same length with ciphertexts) with probability $\frac{1}{2} + \text{Adv}_{\text{ind}}^{\text{PKE-AONT}}$.

Hence, by the discussion above and the triangle inequality,

$$\begin{aligned} |\Pr[S_0]| &= |\Pr[S_0] - \Pr[S_1] + \Pr[S_1]| \\ &\leq |\Pr[S_0] - \Pr[S_1]| + |\Pr[S_1]| \\ &= 2\text{Adv}_{\text{SS}}^{2\text{-PK-TTS}} + \text{Adv}_{\text{ind}}^{\text{PKE-AONT}} + \frac{1}{2}. \end{aligned}$$

Since $\text{Adv}_{\text{SS}}^{2\text{-TTS}}$ and $\text{Adv}_{\text{ind}}^{\text{PKE-AONT}}$ are two negligible functions of λ , we conclude that the advantage of \mathcal{A} winning the semantic security game is bounded by a negligible function of λ .

Theorem 4. *The scheme Π is traceable against t -collusion under the DDH assumption.*

Proof. By contradiction, assume that there exists an adversary \mathcal{A} that, given the public key BK, t of user keys $\{\text{SK}_{u_1}, \text{SK}_{u_2}, \dots, \text{SK}_{u_t}\}$ and an AONT Σ , produces a pirate decoder \mathcal{D} that decrypts all valid ciphertexts perfectly. But when given a probe ciphertext, \mathcal{D} outputs a different value from the pre-computed values in Trace algorithm with non-negligible probabilistic $\epsilon > 0$, i.e., $\Pr[\Sigma(\mathcal{D}(\hat{C}); \rho) \neq M_{k,\sigma}] = \epsilon$. We construct an algorithm \mathcal{B} that breaks the DDH assumption with non-negligible advantage $\frac{\epsilon}{2}$ as follows:

- *Setup.* Algorithm \mathcal{B} is given as input an instance (g, g^u, g^v, X) of DDH assumption, and it wants to determine whether $X = g^{uv}$ or X is a random element in \mathbb{G} (\mathbb{G} has prime order q). \mathcal{B} chooses a position index $k \xleftarrow{\$} \{1, 2, \dots, \ell\}$, chooses $f_j(x) = a_{0,j} + a_{1,j}x \pmod{q}$, where $a_{0,j}, a_{1,j} \xleftarrow{\$} \mathbb{Z}_q^*, \forall j \in \{1, 2, \dots, \ell\}$, chooses $i_0, i_1, z \xleftarrow{\$} \mathbb{Z}_q^*$ and gives \mathcal{A} $\text{BK} = \langle g, (g^{a_{0,j}}, g^{a_{1,j}})_{j=1}^{\ell} \rangle$ but replaces $g^{a_{0,k}}$ by g^u and $g^{a_{1,k}}$ by $(\frac{g^z}{g^u})^{i_{\sigma}^{-1}}$, where $\sigma \xleftarrow{\$} \{0, 1\}$. \mathcal{A} chooses a traitor set $\text{T} \subseteq \{1, \dots, N\}$ of size t to \mathcal{B} . Then \mathcal{B} chooses t codewords $w^{(u_1)}, w^{(u_2)}, \dots, w^{(u_t)} \xleftarrow{\$} \Gamma$ (even if Γ is public, the information of which user get which codeword can be hidden, so \mathcal{B} can choose t codewords by his own) satisfy $w_k^{(u_1)} = w_k^{(u_2)} = \dots = w_k^{(u_t)} = \sigma$ (the existence of these codewords is guaranteed by the fingerprinting codes) and sets \mathcal{A} the keys

$$\begin{aligned} \text{SK}_{u_1} &= \langle w^{(u_1)}, i_0, i_1, (f_1(i_{w_1^{(u_1)}}), \dots, f_{k-1}(i_{w_{k-1}^{(u_1)}}), z, f_{k+1}(i_{w_{k+1}^{(u_1)}}), \dots, f_{\ell}(i_{w_{\ell}^{(u_1)}})) \rangle, \\ \text{SK}_{u_2} &= \langle w^{(u_2)}, i_0, i_1, (f_1(i_{w_1^{(u_2)}}), \dots, f_{k-1}(i_{w_{k-1}^{(u_2)}}), z, f_{k+1}(i_{w_{k+1}^{(u_2)}}), \dots, f_{\ell}(i_{w_{\ell}^{(u_2)}})) \rangle, \\ &\vdots \\ \text{SK}_{u_t} &= \langle w^{(u_t)}, i_0, i_1, (f_1(i_{w_1^{(u_t)}}), \dots, f_{k-1}(i_{w_{k-1}^{(u_t)}}), z, f_{k+1}(i_{w_{k+1}^{(u_t)}}), \dots, f_{\ell}(i_{w_{\ell}^{(u_t)}})) \rangle, \end{aligned}$$

to produces a pirate decoder \mathcal{D} .

- *Trace.* By taking a pirate decoder \mathcal{D} as a decryption oracle, \mathcal{B} runs the modified Trace algorithm as follows:

1. Compute $M \stackrel{\$}{\leftarrow} \Sigma(M'; \rho)$, where $M' \stackrel{\$}{\leftarrow} \mathcal{M}^{\ell'}$, $\rho \stackrel{\$}{\leftarrow} \{0, 1\}^{\tau}$.
2. Choose $j \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$, where $j \neq i_0$ or i_1 . Compute $\bar{c} \leftarrow \langle A_k = m_k X, R = g^v, (j, W_k = X(\frac{(g^v)^z}{X})^{ji_{\sigma}^{-1}}) \rangle$ and set the ciphertext as $\bar{C} \leftarrow \langle k, \bar{c}, Y = \text{MINUS}_k(M) \rangle$.
3. If $\Sigma(\mathcal{D}(\bar{C}); \rho) = \text{COMB}_k(Y, A_k/W_k^{\frac{-i_{\sigma}}{j-i_{\sigma}}} R^{fk(i_{\sigma})\frac{-j}{i_{\sigma}-j}})$, \mathcal{B} answers $X = g^{uv}$ or X is a random element in \mathbb{G} randomly; else \mathcal{B} answers X is a random element in \mathbb{G} .

If $X = g^{uv}$, ciphertext \bar{c} is a valid ciphertext, since

$$X(\frac{(g^v)^z}{X})^{ji_{\sigma}^{-1}} = g^{uv}(\frac{(g^v)^z}{g^{uv}})^{ji_{\sigma}^{-1}} = g^{uv}((\frac{g^z}{g^u})^{i_{\sigma}^{-1}})^{vj} = g^{uv}(g^{a_{1,k}})^{vj} = g^{v(u+a_{1,k}j)}.$$

In this case, $\Sigma(\mathcal{D}(\bar{C}); \rho) = M'_{k,\sigma}$, \mathcal{B} gives the correct answer with probability $\frac{1}{2}$;

If X is a random element in \mathbb{G} , the ciphertext \bar{C} is an invalid ciphertext. In this case, $\Sigma(\mathcal{D}(\bar{C}); \rho) \neq M'_{k,\sigma}$ with probability ϵ , and $\Sigma(\mathcal{D}(\bar{C}); \rho) = M'_{k,\sigma}$ with probability $1 - \epsilon$. Therefore \mathcal{B} gives the correct answer with probability $\epsilon + \frac{1}{2}(1 - \epsilon) = \frac{1}{2} + \frac{\epsilon}{2}$.

Hence, \mathcal{B} solves the DDH problem with non-negligible advantage $\frac{\epsilon}{2}$. This is a contradiction to the DDH assumption. So we conclude that such adversary \mathcal{A} does not exist.

6 Conclusion

We propose a fully-collusion resistant public-key traitor tracing schemes with efficient black-box tracing. It achieves the asymptotically optimal transmission rate for ciphertexts. The storage requirement of each user-key and each public-key are $\ell + 2$ and $2\ell + 1$ respectively, where ℓ is the codeword length of the fingerprinting codes. We show that our scheme is semantically secure based on the DDH hardness assumption and the indistinguishability of the cryptosystem PKE-AONT. Also, our scheme is t -collusion resistant.

There are two open problems. First, How to improve the storage requirements of the user-keys and public-keys further? Second, Billet and Phan [1] proposed a general attack “Pirate 2.0” to attack the code-base traitor tracing schemes. In Pirate 2.0, traitors only give away “part” of user-keys away such that a tracer can trace them with a small probability only. This probability is controlled by traitors according to a specific set of given away keys. How to avoid such attack is also an important problem to make the code-base traitor tracing schemes more practical.

References

1. Billet, O., Phan, D.H.: Traitors collaborating in public: Pirates 2.0. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 189–205. Springer, Heidelberg (2009)

2. Boneh, D., Franklin, M.K.: An efficient public key traitor scheme (Extended abstract). In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)
3. Boneh, D., Naor, M.: Traitor tracing with constant size ciphertext. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM Conference on Computer and Communications Security, pp. 501–510. ACM, New York (2008)
4. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
5. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory* 44(5), 1897–1905 (1998)
6. Boyko, V.: On the security properties of oaep as an all-or-nothing transform. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 503–518. Springer, Heidelberg (1999)
7. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
8. Chabanne, H., Phan, D.H., Pointcheval, D.: Public traceability in traitor tracing schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 542–558. Springer, Heidelberg (2005)
9. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
10. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing traitors. *IEEE Transactions on Information Theory* 46(3), 893–910 (2000)
11. Fazio, N., Nicolosi, A., Phan, D.H.: Traitor tracing with optimal transmission rate. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 71–88. Springer, Heidelberg (2007)
12. Furukawa, J., Attrapadung, N.: Fully collusion resistant black-box traitor revocable broadcast encryption with short private keys. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 496–508. Springer, Heidelberg (2007)
13. Kiayias, A., Yung, M.: Traitor tracing with constant transmission rate. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 450–465. Springer, Heidelberg (2002)
14. Kurosawa, K., Desmedt, Y.G.: Optimum traitor tracing and asymmetric schemes. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 145–157. Springer, Heidelberg (1998)
15. Naor, M., Pinkas, B.: Threshold traitor tracing. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 502–517. Springer, Heidelberg (1998)
16. Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 1–20. Springer, Heidelberg (2001)
17. Phan, D.H.: Traitor tracing for stateful pirate decoders with constant ciphertext rate. In: Nguyễn, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 354–365. Springer, Heidelberg (2006)
18. Phan, D.H., Safavi-Naini, R., Tonien, D.: Generic construction of hybrid public key traitor tracing with full-public-traceability. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 264–275. Springer, Heidelberg (2006)

19. Rivest, R.L.: All-or-nothing encryption and the package transform. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 210–218. Springer, Heidelberg (1997)
20. Tardos, G.: Optimal probabilistic fingerprint codes. In: STOC, pp. 116–125. ACM, New York (2003)
21. Tô, V.D., Safavi-Naini, R., Zhang, F.: New traitor tracing schemes using bilinear map. In: Yung, M. (ed.) Digital Rights Management Workshop, pp. 67–76. ACM Press, New York (2003)
22. Tzeng, W.-G., Tzeng, Z.-J.: A public-key traitor tracing scheme with revocation using dynamic shares. *Des. Codes Cryptography* 35(1), 47–61 (2005)
23. Zhang, R., Hanaoka, G., Imai, H.: On the security of cryptosystems with all-or-nothing transform. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 76–90. Springer, Heidelberg (2004)

Computationally Secure Hierarchical Self-healing Key Distribution for Heterogeneous Wireless Sensor Networks

Yanjiang Yang¹, Jianying Zhou¹, Robert H. Deng², and Feng Bao¹

¹ Institute for Infocomm Research, Singapore

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

² School of Information Systems, Singapore Management University
robertdeng@smu.edu.sg

Abstract. Self-healing group key distribution is a primitive aimed to achieve robust key distribution in wireless sensor networks (WSNs) over lossy communication channels. However, all the existing self-healing group key distribution schemes in the literature are designed for homogenous WSNs that do not scale. In contrast, heterogeneous WSNs have better scalability and performance. We are thus motivated to study self-healing group key distribution for heterogeneous WSNs. In particular, we propose the concept of *hierarchical* self-healing group key distribution, tailored to the heterogeneous WSN architecture; we further revisit and adapt Dutta *et al.*'s model to the setting of hierarchical self-healing group key distribution, and propose concrete schemes that achieve computational security and high efficiency.

Keywords: Heterogeneous wireless sensor network, self-healing group key distribution, scalability.

1 Introduction

Wireless sensor networks (WSNs) have a wide range of potential applications, such as battlefield surveillance, wildlife tracking, healthcare monitoring, and natural disaster monitoring. A WSN consists of a large number of sensor nodes, each being a small sensing device capable of collecting and reporting environmental data to base station. Sensor nodes are extremely constrained in hardware, having limited computation capability, storage capacity, and radio transmission range. Worse yet, sensor nodes are usually powered by batteries, restricted power supply is thus yet another major limitation of WSNs.

As WSNs are often deployed where there is no network infrastructure support, they are easily susceptible to adversaries who can intercept or interrupt the wireless communications. It is thus crucial to ensure secure communication when a WSN is used for mission-critical applications. A fundamental service to achieve secure communication is key distribution, whereby sensor nodes establish (secret) keys, to be used to encrypt and authenticate messages. Unfortunately,

it is commonly acknowledged that key distribution in WSNs is not trivial, considering the resource-constrained nature of sensor nodes. Hence lots of efforts have been dedicated to the study of key management and distribution in WSNs [6,7,9,8,12,15,18,19,20,21,27,28,31]. These methods are categorized into group key distribution [6,9,15,20,21] and pairwise key distribution [7,8,12,18,19,27,31]. The former enables a group of sensor nodes to establish a common group key, while the latter allows pairs of nodes to share distinct keys.

Among the existing group key distribution schemes, self-healing group key distribution [9,21,28] particularly suits WSNs. A prominent property of this type of group key distribution is *self-healing*, which allows group members to recover lost group keys of past sessions based simply on the key update message of the current session. This makes group key distribution resilient to the lossy wireless channels of WSNs. Moreover, self-healing group key distribution offers group member *revocation* such that revoked group members can no longer get the group keys for new sessions after their revocation. This feature is extremely important in mitigating the effect of sensor node compromises: amputate compromised sensor nodes from the WSN, so that the adversary acquiring the secret information of the compromised nodes still cannot get the new group keys.

We observed that all the self-healing group key distribution schemes in the literature considered homogeneous WSNs where all sensor nodes are assumed to be of the same capabilities. However, homogeneous WSNs are not scalable. Indeed, both theoretical and empirical studies have found that the throughput of each sensor node decreases rapidly as the number of nodes increases, and as the traffic becomes heavy, the control overhead due to the underlying routing protocols will consume a large portion of the available bandwidth [11,13]. We are thus motivated to study self-healing group key distribution in *heterogeneous* WSNs. A heterogeneous WSN is composed of not only resource constrained sensor nodes, but also a number of more powerful high-end devices. More specifically, a WSN is partitioned into a number of *groups/clusters*, and a high-end device is placed into each group, acting as the *group manager*/cluster head. Compared to sensor nodes, a group manager is more powerful, and thus does not suffer from the resource scarceness problem as much as a sensor node does.

Our Contributions. Tailored to the heterogeneous WSN architecture, we propose the concept of *hierarchical* self-healing group key distribution. In particular, we formulate a security model for hierarchical self-healing group key distribution by revisiting and adapting Dutta *et al.*'s model [9]. We then propose a basic and an extended scheme, both proven secure under the model. Our construction basically follows Dutta *et al.*'s idea of a combination of a reverse and a forward one-way hash chain, but we show that their model and scheme have some weaknesses, which are rectified in ours. Our extended scheme further exploits the hierarchy of the heterogeneous architecture by secret-sharing the manager key of each group among all group managers, so as to counter possible compromises of group managers. To show that the (extended) scheme is efficient for WSNs, we implement the core (yet the most costly) element of the scheme upon MICAz mote [24], and the experiments demonstrate satisfactory performance.

2 Related Work

Key management and distribution is a security bootstrapping service, fundamental to many other security mechanisms in WSNs, hence tremendous effort has been dedicated to the study of this issue [7,9,8,12,18,19,20,21,27,31]. In general, public key cryptosystems are too expensive for WSNs, and symmetric key primitives such as secret key encryption or cryptographic hash function are often preferred. As such, key management and distribution in WSNs boils down to sharing of secret keys among sensor nodes. To achieve this objective, a commonly used approach is to pre-load a set of secrets inside sensor nodes before their deployment. These pre-loaded secrets are then used either directly as pair-wise keys between each pair of sensor nodes, i.e., pair-wise key distribution [7,8,12,18,19,20,27,31], or as a basis to establish new common keys shared by a group of sensor nodes, i.e., group key distribution [6,9,15,20,21].

Among the existing group key distribution schemes, self-healing group key distribution is particularly suitable for WSNs, because of its self-healing and membership revocation properties. Staddon *et al.* [28] first proposed the concept and a concrete construction of self-healing group key distribution based on secret sharing of two dimensional polynomials. Their construction, however, is not efficient, suffering from high communication and storage overhead. Liu *et al.* [21] then generalized the security notions in [28], and presented a new scheme with better efficiency by combining personal secret distribution with the self-healing technique of [28]. Blundo *et al.* [4] analyzed the security definitions in [21,28] and concluded that it is impossible for any scheme to achieve all of the security requirements formulated in [21,28]. They then formulated a new definition for self-healing group key distribution and came up with a new scheme [5]. Blundo *et al.* [3] also showed an attack to the construction in [28] and discussed the use of randomness in self-healing group key distribution schemes. Other schemes based on the strategy in [28] include [25,14].

All the above self-healing group key distribution schemes are intended to achieve information theoretic security. In [9], Dutta *et al.* proposed novel computationally secure schemes, based on a combination of a reverse one-way hash chain and a forward one-way hash chain. While Dutta *et al.*'s model is weaker and cannot meet all the security requirements put forth in [21,28], their approach tremendously improves the efficiency of the information theoretically secure schemes. Unfortunately, as we shall show in Section 3, Dutta *et al.*'s definition on *the secrecy of personal secrets* in their model has some problems, and we give two attacks on the secrecy of personal secrets in their scheme. The schemes proposed by Du and He [10] followed Dutta *et al.*'s approach, and are also subject to our attacks. Besides rectifying the problems in Dutta *et al.*'s scheme, other differences between our schemes and Dutta *et al.*'s are as follows. First, our schemes are hierarchical, tailored to the heterogeneous WSNs. Second, our schemes achieve authenticated group key distribution, allowing every non-revoked sensor node to verify whether or not its generated group keys are valid, without requiring any extra communications.

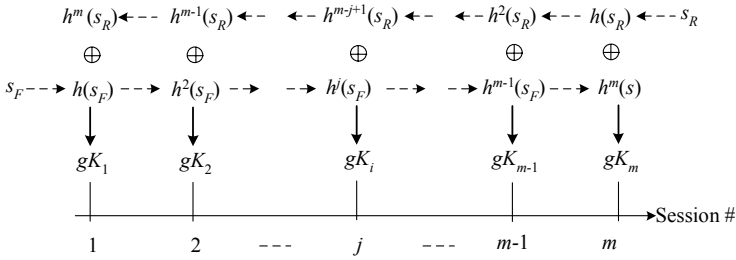


Fig. 1. Definition of Group Keys

3 Review and Analysis of Dutta *et al.*'s Scheme

3.1 Review of Dutta *et al.*'s Scheme

In Dutta *et al.*'s scheme [9], a WSN proceeds in sessions, and at the start of each session the base station broadcasts a key update message to the sensor nodes, enabling the latter to generate a new group key. Suppose the maximal number of sessions supported by the WSN is m . The group key for the j^{th} session is defined as $gK_j = h^j(s_F) + h^{m-j+1}(s_R)$, where h is a one-way hash function, and s_F, s_R are random seeds (see Figure 1). Here, $h^j(.) = \underbrace{h(h(\dots h(.)))}_{j \text{ times}}$. It can be

seen that the hash chain associating with s_R is used in the *reverse* order thus called the reverse hash chain, and that associated with s_F called the forward hash chain. The details of Dutta *et al.*'s scheme are as follows.

- System Initialization. The base station chooses random seeds s_R and s_F , and it also selects m random t -degree polynomials $f_1(x), \dots, f_m(x) \in F_q[x]$, each corresponding to a session, where F_q is a finite field with q being a large prime number, and t is a system parameter denoting the robustness of sensor nodes. The personal secret for a member sensor node i is defined to be $S_i = [f_1(i), \dots, f_m(i)]$. Finally, the base station secretly sends S_i and s_F to each node i .
- Broadcast. At the start of each session, the base station broadcasts a key update message to enable sensor nodes to generate a new group key. Let $R_j = \{i_1, \dots, i_w\}$ be the set of revoked sensor nodes upon the start of session $j \in \{1, \dots, m\}$ and $|R_j| = w \leq t$. The base station computes the following polynomials:

$$\begin{aligned}
 r_j(x) &= (x - i_1) \cdots (x - i_w) \\
 b_j(x) &= h(s_R)^{m-j+1} \cdot r_j(x) + f_j(x),
 \end{aligned} \tag{1}$$

where $r_j(x)$ is called the revocation polynomial. Finally, the base station broadcasts the key update message B_j to all sensor nodes, where $B_j = R_j \cup \{b_j(x)\}$.

- Session Key Generation. Upon receipt of B_j , if node i is not revoked, it is able to recover $h^{m-j+1}(s_R) = \frac{b_j(i) - f_j(i)}{r_j(i)}$. Then it continues to compute the group key $gK_j = h^{m-j+1}(s_R) + h^j(s_F)$ using s_F .
- Addition of New Group Member. A newly added member in session j is not allowed to compute group keys of previous sessions. To add a new member with ID α starting from session j , the group manager computes and gives $S_\alpha = \{f_j(\alpha), f_{j+1}(\alpha), \dots, f_m(\alpha)\}$ and $h^j(s_F)$ to the node.

3.2 Attacks

In Dutta *et al.*'s model [9], the secrecy of personal secrets is defined as *any t or less revoked members cannot compute the personal secrets of other members*. We next give two attacks, showing that their scheme cannot achieve the secrecy of personal secrets. We also notice that the self-healing key distribution scheme proposed by Du and He in [10] follow Dutta *et al.*'s approach, and our attacks apply to their scheme too (to avoid repetition, we do not review their scheme).

Attack 1. In the above construction, the revocation polynomial $r_j(x)$ is simply defined as $r_j(x) = (x - i_1) \cdots (x - i_w)$. Let $f_j(x) = a_t x^t + \cdots + a_1 x + a_0$. It is clear that broadcasting $b_j(x)$ directly reveals a_t, a_{t-1}, \dots , and a_{w+1} . This means that $f_j(x)$ only has $w + 1 \leq t$ unknown coefficients, i.e., a_w, \dots, a_0 , and any $w + 1$ (instead of $t + 1$) revoked nodes together can determine $f_j(x)$ and in turn compute $f_j(i)$ for any i . Therefore, the scheme cannot achieve the secrecy of personal secrets.

Attack 2. Let us consider a particular non-revoked node i in session j . From the broadcast message B_j , node i calculates $h(s_R)^{m-j+1} = \frac{b_j(i) - f_j(i)}{r_j(i)}$. Based on $h(s_R)^{m-j+1}$, node i can actually compute any $f(i'), i' \neq i$, as $f(i') = b_j(i') - h(s_R)^{m-j+1} \cdot r_j(i')$. This suggests that once the group key for a session is established, the element of a sensor node's personal secret corresponding to that session is revealed to all other non-revoked nodes. As such, even node i is revoked in a subsequent session, it already knows a part (albeit corresponding to the past sessions) of other non-revoked nodes' personal secrets.

4 Model and Security Definition

4.1 Heterogeneous Architecture

We consider the heterogeneous architecture, where a WSN is partitioned into a number of *groups*. A high-end device is placed into each group, acting as the *group manager*. Compared to sensor nodes, high-end group managers have relatively higher computation capability, larger storage size, and longer radio range. They also have longer power supply, and can even be line-powered in some circumstances, e.g., when a WSN is deployed to monitor a building, the group managers can easily tap on the electricity lines to get power supply. Therefore

unlike sensor nodes, group managers do not suffer too much from the resource scarceness problem. Depending on applications, hardware capabilities of a group manager may vary from that comparable to a bluetooth device to that of a high end PDA. The introduction of high-end group managers into a WSN makes the once homogeneous network *heterogeneous*. The entire network including base station, group managers, and sensor nodes forms a logically hierarchical architecture, as depicted in Figure 2.

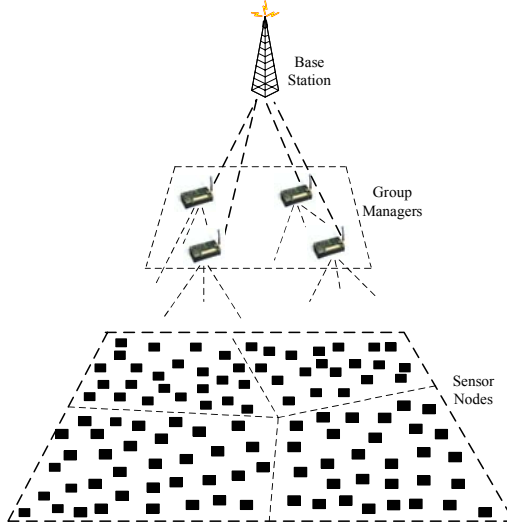


Fig. 2. Heterogeneous WSN Architecture

In this architecture, downlink messages broadcast by the base station directly reach sensor nodes, whereas uplink messages sent by a sensor node to the base station is forwarded via its group manager, which acts as an intermediary between the base station and the sensor nodes within its jurisdiction. A sensor node may reach the group manager directly, or by traversing a *short* multi-hop path. Intuitively, the inclusion of powerful group managers provides shortcuts for data delivered from the sensor nodes to the base station, so the overall system performance and in turn the lifetime of the network are expected to be greatly improved. Indeed, numerous studies have corroborated the higher efficiency of the heterogeneous WSN architecture, e.g., [16,26,29].

4.2 System Model

Three types of entities are involved in our hierarchical group key distribution system: base station, group managers, and a large number of sensor nodes. The sensor nodes are partitioned into a number of N_G groups, and each group G_ℓ has a group manager, $\ell \in \{1, \dots, N_G\}$. Each sensor node in a group is uniquely identified by an ID number i , where $i \in I_\ell \subseteq \{1, \dots, n\}$, where I_ℓ is the set of all node ID numbers in G_ℓ and n is the largest possible ID number in the system.

Corresponding to the heterogenous architecture, the keys held by the entities form a hierarchy, as shown in Figure 3: the base station holds a *root key* at level 2, each group manager has a distinct *manager key* at level 1, and at level 0 sensor nodes in each group hold a *group key* during each session. A key at a lower level is generated from the keys at higher levels, but not the other way around. This key hierarchy helps to implement “separation of duty” within the system, e.g., it is not necessary for the sensor nodes to process the control messages broadcast by the base station to the group managers.

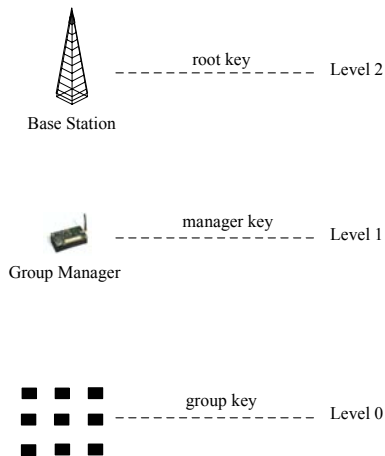


Fig. 3. Key Hierarchy

A group manager takes charge of distribution of group keys within its group. A group key is uniquely associated with a session. To distribute a group key for a new session, the group manager broadcasts a *key update message* to all its sensor nodes. The group key is then computed by a sensor node based on the received key update message and its preloaded *personal secret*. Denote the personal secret of sensor node i as S_i , which is a vector of m elements with m being the maximum number of sessions. Each element in S_i corresponds to a session and we use $S_i[j]$ to denote the element corresponding to the j th session, $j \in \{1, \dots, m\}$. $S_i[j]$ becomes *obsolete* once the group key for the j th session is established; otherwise $S_i[j]$ is *fresh*. A sensor node can be revoked or non-revoked, and only non-revoked sensor nodes can compute the group keys.

4.3 Adversary Model

As usual, we assume that the base station is trusted. In our basic scheme, the group managers are also presumed trusted, but this assumption is removed in the extended scheme. We are mainly concerned with the distribution of group keys among sensor nodes. As such, we assume an adversary is able to passively eavesdrop on, or actively intercept, modify, insert, or drop key update messages

from a group manager to all its sensor nodes. We also allow the adversary to compromise up to t sensor nodes in a group, where t is a system parameter.

4.4 Security Definition

We formally define the concept and security requirements of hierarchical self-healing group key distribution, by revisiting and extending the definition in [9].

Definition 1. (Hierarchical Self-healing Group Key Distribution with t -Revocation) *Let N_G, n, m, t be system parameters defined as above. \mathcal{D} is hierarchical self-healing group key distribution with t -revocation, if the following holds:*

- a. (Key Hierarchy) *The manager keys held by the group managers are derived from the root key of the base station, but it is computationally infeasible to compute the root key from the manager keys. The same relationship should hold between group keys and the corresponding manager key.*
- b. (Secrecy of Personal Secret) *For any $U_\ell \subset \{1, \dots, n\}$ in group $G_\ell, \ell \in \{1, \dots, N_G\}$, if $|U_\ell| \leq t$, then it is computationally infeasible for the nodes in U_ℓ to collectively determine the fresh elements of S_i for any $i \notin U_\ell$.*
- c. (Authenticated Generation of Group Key) *Let $gK_{\ell,j}$ be the group key of group G_ℓ for session j , and $B_{\ell,j}$ be the broadcast key update message from the group manager, where $j \in \{1, \dots, m\}$. For any non-revoked sensor node i in the group, $gK_{\ell,j}$ is efficiently computed from $B_{\ell,j}$ and $S_i[j]$ in an authenticated manner. On the contrary, it is computationally infeasible to compute $gK_{\ell,j}$ from the key update message or a personal secret alone.*
- d. (t -Revocation) *For any session j , let $R_{\ell,j}$ be the set of revoked nodes in G_ℓ at the start of session j , where $|R_{\ell,j}| \leq t$, it is computationally infeasible to compute $gK_{\ell,j}$ from the broadcast message $B_{\ell,j}$ and $\{S_i\}_{i \in R_{\ell,j}}$.*
- e. (t -wise Forward Secrecy) *Let $U_\ell \subseteq \{1, \dots, n\}$ denote the sensor nodes which joined G_ℓ after session j . Given that $|U_\ell| \leq t$, it is computationally infeasible for all members in U_ℓ to collectively compute $gK_{\ell,1}, \dots, gK_{\ell,j}$, even with the knowledge of $gK_{\ell,j+1}, \dots, gK_{\ell,m}$.*
- f. (Self-healing) *A non-revoked sensor node in G_ℓ between sessions j_1 and j_2 , $1 \leq j_1 < j_2 \leq m$, can efficiently compute any $gK_{\ell,j}$, $j_1 \leq j \leq j_2$, from B_{ℓ,j_2} and its personal secret.*

Remark. Compared to Dutta *et al.*'s model, we distinguish between *obsolete elements* and *fresh elements* of a personal secret, and the secrecy of personal secret in our definition actually mandates the secrecy of fresh elements. This differentiation addresses our second attack, and suggests that personal secret should not be used for purposes other than distribution of group keys, and once an element is obsolete, it should be discarded immediately.

5 Basic Scheme

We first present a basic hierarchical self-healing group key distribution scheme, assuming that the group managers are trusted. We suppose that the set of

revoked users is monotonic, i.e., a revoked user never rejoins the network. Let F_q be a finite field, where q is a large prime number. All arithmetic operations below are performed in F_q . Let $h, h_R, h_F : \{0, 1\}^* \rightarrow F_q$ be cryptographic hash functions, and N_G, n, m, t be system parameters. The basic scheme is a slight extension of Dutta *et al.*'s scheme reviewed above, and rectifies its weaknesses.

- **System Initialization.** The base station chooses a root key $rK = [rk_1, rk_2]$, where rk_1 and rk_2 are random numbers of appropriate length. For each group $G_\ell, \ell = \{1, \dots, N_G\}$, the base station computes a manager key as $mK_\ell = [mk_{\ell,1}, mk_{\ell,2}]$, where $mk_{\ell,1} = h(G_\ell, rk_1)$ and $mk_{\ell,2} = h(G_\ell, rk_2)$. Clearly, it is computationally infeasible to compute the root key from the manager keys. Then the base station securely passes the manager keys to the corresponding group managers. We do not specify how this can be done, but it often suffices by using some out-of-band channel.

Upon receipt of the manager keys, the group managers begin the preparation for setting up group keys. In particular, the group manager for G_ℓ whose manager key is $mK_\ell = [mk_{\ell,1}, mk_{\ell,2}]$ sets $mk_{\ell,1}$ to be the seed $s_{\ell,R}$ of the reverse one-way hash chain of length $m + 1$:

$$\begin{aligned} k_{\ell,R}^j &= h_R(k_{\ell,R}^{j-1}) \\ &= h_R(h_R(k_{\ell,R}^{j-2})) = \dots \\ &= h_R^j(s_{\ell,R}), 1 \leq j \leq m + 1 \end{aligned} \quad (2)$$

and sets $mk_{\ell,2}$ to be the seed $s_{\ell,F}$ for the forward hash chain of length m :

$$\begin{aligned} k_{\ell,F}^j &= h_F(k_{\ell,F}^{j-1}) \\ &= h_F(h_F(k_{\ell,F}^{j-2})) \dots \\ &= h_F^j(s_{\ell,F}), 1 \leq j \leq m \end{aligned} \quad (3)$$

The group key $gK_{\ell,j}$ for session $j \in \{1, \dots, m\}$ is defined to be $gK_{\ell,j} = k_{\ell,R}^{m-j+1} + k_{\ell,F}^j = h_R^{m-j+1}(s_{\ell,R}) + h_F^j(s_{\ell,F})$. The group manager next selects m random t -degree polynomials $f_{\ell,1}(x), \dots, f_{\ell,m}(x) \in F_q[x]$, each corresponding to a session. The personal secret for the member sensor node i is defined to be $S_i = [f_{\ell,1}(i), \dots, f_{\ell,m}(i)]$. The group manager then sends $S_i, k_{\ell,R}^{m+1}$ and $s_{\ell,F}$ to each node i in a secure manner, e.g., preloading before the deployment of nodes. Note that $k_{\ell,R}^{m+1}$ will be used as the initial *authenticator*, denoted as I_ℓ , in subsequent group key generation process.

- **Broadcast.** The broadcast procedure for EACH group is almost the same as in Dutta *et al.*'s scheme, with the main exception on the computation of the revocation polynomial. In particular, let $R_{\ell,j} = \{i_1, \dots, i_w\}$ be the set of revoked sensor nodes in G_ℓ upon the start of session $j \in \{1, \dots, m\}$ and $|R_{\ell,j}| = w \leq t$. The group manager chooses a random set $R'_{\ell,j} = \{i'_t, \dots, i'_{w+1}\} \subset \{1, \dots, n\} \setminus I_\ell$, where I_ℓ is the set of all node IDs in G_ℓ . That is, the group manager chooses $t - w$ random IDs that are not in that group. Then, the revocation polynomial $r_{\ell,j}(x)$ is computed as $r_{\ell,j}(x) =$

$(x - i_1) \cdots (x - i_w)(x - i'_{w+1}) \cdots (x - i'_t)$. It is clear that $r_{\ell,j}(x)$ defined as such avoids our first attack. $b_{\ell,j}(x)$ is then computed with this $r_{\ell,j}(x)$ as in Dutta *et al.*'s scheme (Eqn. [10](#)). Accordingly, the key update message $B_{\ell,j}$ also includes $R'_{\ell,j}$, i.e., $B_{\ell,j} = R_{\ell,j} \cup R'_{\ell,j} \cup \{b_{\ell,j}(x)\}$.

- **Session Key Generation.** The main difference from that in Dutta *et al.*'s scheme is that each sensor node in G_ℓ holds an authenticator Γ_ℓ . As such, when a non-revoked node recovers $k_{\ell,R}^{m-j+1} = \frac{b_{\ell,j}(i) - f_{\ell,j}(i)}{r_{\ell,j}(i)}$, it validates $k_{\ell,R}^{m-j+1}$ using Γ_ℓ : for example, if $\Gamma_\ell = k_{\ell,R}^{m+1}$ (the initial value [11](#)), then the validation is to test $\Gamma_\ell \stackrel{?}{=} h_R^j(k_{\ell,R}^{m-j+1})$. Other steps remain the same as in Dutta *et al.*'s scheme.
- **Addition of New Group Member.** This procedure is the same as in Dutta *et al.*'s scheme.

Efficiency. This scheme is highly efficient in terms of storage, communication, and computation overhead. For storage, the personal secret together with the authenticator accounts for $(m+1) \log q$ bits storage in each sensor node (compared to Dutta *et al.*'s scheme, ours only needs $\log q$ -bit more storage for the authenticator). For communications, our scheme generates $t(\log q + \log n) \approx t \log q$ bits key update message (since $n \ll q$), which is almost the same as the bit length of the key update message in Dutta *et al.*'s scheme. For computation, no costly public key primitive is involved in our scheme, and the computation overhead inflicted upon sensor nodes includes only cryptographic hash function and polynomial operations.

Security. For security of the scheme, we have the following theorem and the proof can be found in [30](#).

Theorem 1. *The above construction is a hierarchical self-healing group key distribution scheme with respect to Definition 1.*

6 Extended Scheme

In the above basic scheme, we assumed that the group managers are trusted and not compromised. We next deal with the potential compromises of group managers in our system. There are two aspects to this problem: how to timely detect break-ins to group managers and how to mitigate the damage once a group manager is compromised. The first aspect can be addressed along the lines of cooperative intrusion detection techniques as proposed in [21722](#). We next discuss to address the second aspect, and in particular how to mitigate the damages caused by group managers' compromises.

It should be clear that once a group manager is compromised, the manager key it holds is unavoidably disclosed. In the basic scheme, since the manager key is used to derive all group session keys, once the manager key is revealed, so does all

¹ For better efficiency, a node should overwrite Γ_ℓ by setting $\Gamma_\ell = k_{\ell,R}^{m-j+1}$ at the end of the session key generation procedure.

the derived group keys. As such, probably the best we can expect is that *in case a group manager is compromised in a certain session, only the group keys for that and earlier sessions are revealed, without affecting subsequent sessions*. Note that since we desire self-healing, it is inevitable that disclosure of the manager key in a session leads to the disclosure of the group keys for all earlier sessions. To this end, the manager key of a group should be *sessional* too, rather than a constant quantity as in the basic scheme². According to this rationale, we propose the following approaches to address the issue of compromises of group managers.

6.1 Naive Approach

A naive approach is to involve the base station into distributing the sessional manager keys to the group managers. Specifically, at the initialization phase, the base station generates $s_{\ell,R}$ and $s_{\ell,F}$ for each group G_ℓ using the root key, as per the basic scheme, i.e., $s_{\ell,R} = h(G_\ell, rk_1)$ and $s_{\ell,F} = h(G_\ell, rk_2)$. However, the base station does not pass $s_{\ell,R}$ to the group manager, but keeps it to itself. For each session $j \in \{1, \dots, m\}$, the base station computes $k_{\ell,R}^{m-j} = h_R^{m-j}(s_{\ell,R})$ (Eqn. [2](#)) and sends it to the corresponding group manager. Here we assume a secure communication channel between the base station and each group manager. Then, the sessional manager key held by the group manager is $mK_{\ell,j} = [k_{\ell,R}^{m-j}, s_{\ell,F}]$. Using $k_{\ell,R}^{m-j}$, the group manager computes and broadcasts $k_{\ell,R}^{m-j+1} = h_R(k_{\ell,R}^{m-j})$ to the sensor nodes as in the basic scheme (Eqn. [1](#)), which enables the nodes to compute group key $gK_{\ell,j} = k_{\ell,R}^{m-j+1} + k_{\ell,F}^j$. It is easy to see that the manager keys thus generated sustain key hierarchy, and compromise of $mK_{\ell,j}$ does not disclose group keys for sessions later than $j + 1$. This is *almost* the best we can expect.

This approach is quite simple. However, the involvement of the base station offsets the benefits offered by the heterogeneous architecture, one of which is to dispense with the implication of the base station into the management (including security enforcement) of individual groups. The involvement of the base station may result in single point of failure, in the sense that once the adversary manages to block the base station by, e.g., DoS attacks, the whole system is crashed.

6.2 An Extended Scheme

The extended scheme tries not to get the base station involved, as in the basic scheme. As such, secret-sharing $s_{\ell,R}$ of each group among multiple group

² One may argue that since the group manager is the only party that takes charge of the distribution of group session keys, once the group manager is compromised, all subsequent sessions of that group will fall to the control of the adversary, regardless of the measures taken to protect the manager key. This argument is actually based on the assumption that the adversary can continue controlling the compromised group manager. We, however, expect that the base station can timely detect the compromised group manager (e.g., using the cooperative intrusion detection techniques) and recover it in a certain later session. In fact, we believe that a WSN in real application should achieve this. Our extended schemes aim to work in such a scenario.

managers seems to be the only possible solution. The approach should satisfy that at no time should $s_{\ell,R}$ be reconstructed at any group manager including the one it belongs to. This requires computing $k_{\ell,R}^j, j = 1, \dots, m$, in a distributed way, while without reconstructing $s_{\ell,R}$. It is clear that if $h_R()$ is a regular cryptographic one-way hash function, it is quite hard (if possible) to compute $k_{\ell,R}^j = h_R^j(s_{\ell,R})$ as required. We have to find a special $h_R()$ that facilitates the desired distributed computation.

Our choice for $h_R()$ is $h_R : Z_N \rightarrow QR_N$, and in particular $h_R(x) = x^2 \pmod{N}$, where N is a product of two large primes such that factorization of N is computationally intractable, and x 's are sufficiently large numbers in Z_N . In such a case, $h_R(\cdot)$ is a one-way function, under the intractability assumption of computing square root modulo a composite. The well known Rabin encryption is based on this assumption. $h_R(\cdot)$ thus defined has the following property: suppose $s_{\ell,R}$ has t' multiplicative shares $\pi_1, \dots, \pi_{t'}$ such that $s_{\ell,R} = \pi_1 \times \dots \times \pi_{t'} \pmod{N}$, then $h_R^j(s_{\ell,R}) = (s_{\ell,R})^{2^j} = (\pi_1)^{2^j} \times \dots \times (\pi_{t'})^{2^j} \pmod{N}$. This property well meets our need of computing $h_R^j(s_{\ell,R})$ without reconstructing $s_{\ell,R}$. By using this $h_R(\cdot)$, we next highlight the main idea of our scheme. The communication channel among the group managers is assumed secure. Let us further suppose that we offer t' -robustness to our system, i.e., the adversary does not recover $s_{\ell,R}$ unless compromising t' or more group managers, where t' is a system parameter.

- **System Initialization.** The base station selects the root key and computes $s_{\ell,R}$ and $s_{\ell,F}$ for each group G_ℓ , as in the basic scheme. To secret-share $s_{\ell,R}$ among all group managers, the base station partitions $s_{\ell,R}$ into t' shares $\pi_1, \dots, \pi_{t'}$ such that $s_{\ell,R} = \pi_1 \times \dots \times \pi_{t'} \pmod{N}$; then securely sends π_1 together with $s_{\ell,F}$ to the group manager of G_ℓ , and sends each of the remaining shares to $\lfloor (N_G - 1)/(t' - 1) \rfloor$ other group managers. That is, the shares, except the one held by the group manager of G_ℓ , are evenly distributed among the remaining $N_G - 1$ group managers. Note that this offers resilience to the share availability, in the sense that loss of some shares does not affect computation of the manager keys. This also gives a higher weight to π_1 held by the group manager of G_ℓ , without which the group session keys cannot be computed.

The steps taken by the group managers in preparation for setting up group keys are the same as in the basic scheme.

- **Broadcast.** In session j , the group manager of G_ℓ asks other group managers to help generate $k_{\ell,R}^{m-j} = h_R^{m-j}(s_{\ell,R})$ as follows: each group manager raises the share π_i at its disposal to the power of 2^{m-j} , i.e., $\zeta_i = \pi_i^{2^{m-j}}$, and passes the result to the group manager of G_ℓ , who then computes $k_{\ell,R}^{m-j}$ by pooling (multiplying) together a combination of appropriate ζ_i 's (including its own). The pooling procedure has to filter out redundant and erroneous shares. We stress that there are many means to detect erroneous shares, e.g., in the initialization phase, the base station gives the group manager of G_ℓ an *authenticator* for each share. The manager key $mK_{\ell,j}$ is then set to

$mK_{\ell,j} = [k_{\ell,R}^{m-j}, s_F]$. The remaining steps are the same as in the above naive approach.

- **Session Key Generation & Addition of New Member.** These steps remain the same as in the basic scheme and the naive approach.

Security: Security of this approach is straightforward, given the security of the basic scheme. We only point out that in this approach, t -revocation is based on the intractability assumption of computing square root modulo a composite, instead of the one-way-ness of the cryptographic hash function.

Experimental Results: In both of the basic and the extended schemes, the bit length of q should be at least equal to that of $h_R(\cdot)$, i.e., $|q| \geq |h_R(\cdot)|$. For the basic scheme, since $h_R(\cdot)$ can be instantiated by a regular cryptographic hash function, it suffices that $|q| = 161$, assuming $|h_R(\cdot)| = 160$. However, in the extended scheme, $|h_R(\cdot)| = |N|$. To make factorization of N hard, $|N|$ should be at least 1024 bits, so should be $|q|$. We thus need to examine the actual efficiency of the extended scheme.

For computation overhead, squaring operations (i.e., computing $h_R(\cdot)$) dominate the workload of sensor nodes. Note that although a squaring operation is an exponentiation, it is essentially also a multiplication operation. Thus in principle, squaring operations are not deemed expensive. Nevertheless, as squaring operations are the most costly part in our scheme, it still makes sense to gauge their actual computation cost on real world sensors. To this end, we tested upon MICAz mote [24] running TinyOS 2.0. Hardware configuration of MICAz mote includes a ATmega 128L (8-bit,8MHz) processor, 128K-byte program memory, and 2.4 GHz radio transmission. Figure 4 lists the experimental results on the timing of squaring operations over x 's of varying sizes, modulo a 1024-bit N . The

Size of x (bits)	Timing (milliseconds)	Size of x (bits)	Timing (milliseconds)
650	26	850	59.6
700	32.5	900	71
750	39.7	950	83.2
800	51.3	1000	98.6

Fig. 4. Experimental Results

results indicate that a squaring operation takes less than 100 milliseconds, which is quite satisfactory for WSNs. For storage overhead, each sensor node needs to store the personal secret, each element of which is $|q| = 1024$ bits. Recall that the MICAz mote we experimented upon has 128K-byte program memory. Even the mote allocates 8K-byte program memory to store personal secrets, the network can have approximately 60 sessions.

7 Conclusion

Both experimental and theoretical studies have shown that heterogeneous WSNs have better scalability and performance than homogenous ones. However, all existing self-healing group key distribution schemes consider homogenous WSNs. We were thus motivated to study hierarchical self-healing group key distribution for heterogeneous WSNs. In particular, we formulated a model for hierarchical self-healing group key distribution, and proposed concrete schemes that achieve computational security (instead of information theoretic security as in previous schemes in the literature) and high efficiency.

Acknowledgement

This work is supported by the A*STAR project SEDS-0721330047.

References

1. Blundo, C.: Randomness in Self-healing Key Distribution Schemes. In: Proc. IEEE Information Theory Workshop, pp. 80–84 (2005)
2. Buchegger, S., Le Boudec, J.: Performance Analysis of the CONFIDANT Protocol Cooperation of Nodes - Fairness in Dynamic Ad-hoc Networks. In: Proc. 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2002, pp. 226–236 (2002)
3. Blundo, C., D'Arco, P., Listo, M.: A Flaw in A Self-healing Key Distribution Schemes. In: Proc. Information Theory Workshop, pp. 163–166 (2003)
4. Blundo, C., D'Arco, P., Santis, A., Listo, M.: Definitions and Bounds for Self-healing Key Distribution. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 234–245. Springer, Heidelberg (2004)
5. Blundo, C., D'Arco, P., Santis, A., Listo, M.: Design of Self-healing Key Distribution Schemes. Designs, Codes and Cryptography 32(1-3), 15–44 (2004)
6. Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly-secure key distribution for dynamic conferences. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 471–486. Springer, Heidelberg (1993)
7. Chan, H., Perrig, A., Song, D.: Random Key Pre-distribution Schemes for Sensor Networks. In: Proc. IEEE Symposium on Security and Privacy, pp. 197–213 (2003)
8. Du, W.L., Deng, J., Han, Y.S., Varshney, P.K.: A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In: Proc. ACM Conference on Computer and Communication Security, CCS 2003, pp. 42–51 (2003)
9. Dutta, R., Change, E.C., Mukhopadhyay, S.: Efficient Self-healing Key Distribution with Revocation for Wireless Sensor Networks Using One Way Key Chains. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 385–400. Springer, Heidelberg (2007)
10. Du, W., He, M.: Self-healing Key Distribution with Revocation and Resistance to the Collusion Attack in Wireless Sensor Networks. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 345–359. Springer, Heidelberg (2008)
11. Das, S., Perkins, C., Royer, E.: Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In: Proc. of IEEE INFOCOM 2000, vol. 1, pp. 3–12. IEEE Press, Los Alamitos (2000)

12. Eschenauer, L., Gligor, V.D.: A Key-Management Scheme for Distributed Sensor Networks. In: Proc. ACM Conference on Computer and Communication Security, CCS 2002 (2002)
13. Gupta, P., Kumar, P.: The Capacity of Wireless Networks. *IEEE Transactions on Information Theory* 46(2), 388–404 (2000)
14. Hong, D., Kang, J.: An Efficient Key Distribution Scheme with Self-healing Property. *IEEE Communication Letters* 9, 759–761 (2005)
15. Huang, D., Mehta, M., Medhi, D., Harn, L.: Location-aware Key Management Scheme for Wireless Sensor Networks. In: Proc. 2nd ACM workshop on Security of Ad Hoc and Sensor Networks
16. <http://www.intel.com/research/exploratory/heterogeneous.htm>
17. Kachirski, O., Guha, R.: Effective Intrusion Detection Using Multiple Sensors in Wireless Ad Hoc Networks. In: Proc. 36th Annual Hawaii International Conference on System Sciences, HICSS 2003 (2003)
18. Liu, D., Ning, P.: Location-based Pairwise Key Establishment for Relatively Static Sensor Networks. In: Proc. ACM Workshop on Security of Ad hoc and Sensor Networks (2003)
19. Liu, D., Ning, P.: Improving Key Pre-distribution with Deployment Knowledge in Static Sensor Networks. *ACM Transactions on Sensor Networks* (2005)
20. Liu, D., Ning, P., Du, W.L.: Group-based Key Pre-distribution in Wireless Sensor Networks. In: Proc. ACM Workshop on Wireless Security (2005)
21. Liu, D., Ning, P., Sun, K.: Efficient Self-Healing Group Key Distribution with revocation Capability. In: Proc. ACM Conference on Computer and Communication Security, CCS 2003 (2003)
22. Marchang, N., Datta, R.: Collaborative Techniques for Intrusion Detection in Mobile Ad-hoc Networks. *Ad Hoc Network* 6, 508–523 (2008)
23. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In: Proc. 5th Annual Symposium on Operating Systems Design and Implementation, OSDI 2002 (2002)
24. <http://www.xbow.com/Products/productdetails.aspx?sid=156>
25. More, S., Malkin, M., Staddon, J.: Sliding-window Self-healing Key Distribution with Revocation. In: Proc. ACM Workshop on Survivable and Self-regenerative System (2003)
26. Mache, J., Wan, C.Y., Yarvis, M.: Exploiting heterogeneity for sensor network security. In: Proc. IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2008, pp. 591–593 (2008)
27. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, D.: SPINS: Security Protocols for Sensor Networks. *Wireless Networks Journal, WINE* (2002)
28. Staddon, J., Miner, S., Franklin, M., Balfanz, D., Malkin, M., Dean, D.: Self-healing Key Distribution with Revocation. In: Proc. IEEE Symposium on Security and Privacy, S&P 2002, pp. 241–257 (2002)
29. Yarvis, M., et al.: Exploiting Heterogeneity in Sensor Networks. In: Proc. IEEE INFOCOM 2005 (2005)
30. Yang, Y.J., Zhou, J.Y., Deng, R.H., Bao, F.: Hierarchical Self-Healing Key Distribution for Heterogeneous Wireless Sensor Networks. In: Proc. Securecomm 2009 (2009),
<http://icsd.i2r.a-star.edu.sg/staff/yanjiang/papers/securecomm09.pdf>
31. Zhu, S., Setia, S., Jajodia, S.: LEAP: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks. In: Proc. ACM Conferenc on Computer and Communication Security, CCS 2003, pp. 62–72 (2003)

Enabling Secure Secret Updating for Unidirectional Key Distribution in RFID-Enabled Supply Chains

Shaoying Cai¹, Tieyan Li², Changshe Ma¹, Yingjiu Li¹, and Robert H. Deng¹

¹ School of Information Systems, Singapore Management University
shaoyingcai.2009@phdis.smu.edu.sg,

{changshema,yjli,robertdeng}@smu.edu.sg

² Institute for Infocomm Research, A*STAR Singapore
litieyan@i2r.a-star.edu.sg

Abstract. In USENIX Security 08, Juels, Pappu and Parno proposed a secret sharing based mechanism to alleviate the key distribution problem in RFID-enabled supply chains. Compared to existing pseudonym based RFID protocols, the secret sharing based solution is more suitable for RFID-enabled supply chains since it does not require a database of keys be distributed among supply chain parties for secure ownership transfer of RFID tags. However, this mechanism cannot resist tag tracking and tag counterfeiting attacks in supply chain systems. It is also not convenient for downstream supply chain parties to adjust the size of RFID tag collections in recovering tag keys. To address these problems, we propose a flexible and secure secret update protocol which enables each supply chain party to update tag keys in a secure and efficient manner. Our proposal enhances the previous secret sharing based mechanism in that it not only solves the flexibility problem in unidirectional key distribution, but also ensures the security for ownership transfer of tags in RFID-enabled supply chains.

1 Introduction

Radio-frequency identification (RFID) is a wireless Automatic Identification and Data Capture (AIDC) technology that has been widely deployed in many applications including supply chain management. While RFID technology facilitates efficient management of RFID tags, it also triggers privacy concerns since sensitive information about RFID tags may be collected by an adversary via wireless communication channel without their owner's awareness. To prevent RFID tags from unwanted readouts, various solutions have been proposed in the literature. One solution is to send a *Kill* command [2] to a tag so that the tag is "dead" to any queries. Another solution is to physically clip a tag's antenna so that the tag is silenced [7]. Juels, Rivest and Szyldo introduced the blocker tag [6], which generates a signal that collides with all tags to be protected. Once the blocker tag is removed or disabled, however, the privacy of the protected tags may be disclosed. To make RFID tags always readily responsible, Fishkin, Roy

and Jiang [3] proposed a distance-based access control scheme in which the tags reply with different levels of details depending on their distance to the reader, assuming that an adversary cannot get close enough to a tag as an authorized reader. However, it is difficult to prevent an adversary from getting close to a tag in many applications; consequently, the adversary may obtain sensitive information about some tags.

To address this concern, many protocols based on pseudonymous ID have been proposed [5], including the hash-lock of Weis et al. [15], the hash chain of Ohkubo, Suzuki and Kinoshita [12], the tree-based proposal of Molnar and Wagner [11], and many randomized pseudonym-based protocols such as the one proposed by Li and Ding [10]. In such protocols, each tag is associated with a pseudonymous ID. An authorized reader can identify the tag from its pseudonyms because it has access to a database that maps each tag's pseudonym to its real ID based on a secret key. Without the database, an adversary cannot identify a tag nor track it from its pseudonyms (which may change from time to time). When the tagged items change their ownership in a supply chain, the database needs to be transferred so that the new owner can recognize the tags according to the database. How to efficiently distribute the database among authorized parties is critical for applying pseudonym based protocols in RFID-enabled supply chains. Since the database consists of the keys for all tags, this problem is essentially a key distribution problem to be addressed in RFID-enabled supply chains.

In supply chain practice, especially for dynamic or ad hoc supply chain structure, the parties in supply chain are usually lack of secure network connections. A practical solution to the key distribution problem is to split a tag key into a number of shares and store the shares to multiple tags. Since the tag keys are stored in the tags directly, an authorized reader/party can collect enough shares and recover the keys, while an adversary is assumed to have limited access to the tags such that he/she cannot collect enough shares for recovering the keys. In this solution, there is no need of distributing a key database among supply chain parties. This secret sharing based solution is thus particularly useful for protecting dynamic and ad hoc supply chains.

A recent work in this direction is conducted by Juels, Pappu and Parno [4], which we call Juels-Pappu-Parno key sharing mechanism, or JPP mechanism for short. In this solution, a common key k of a batch of tags is split with a (τ, n) -secret sharing scheme, and each tag T_i stores a share S_i of k and its individual (encrypted) information M_i . A reader can recover k with access to at least τ shares. From k and M_i , a reader can decrypt the information about tag T_i . The reader is referred to Section 2.3 for more details about the JPP mechanism. This proposal does not restrict on the time period in which a tag should be read, or the number of the tags which should be attached to an item; thus, it is considered to be the only secret sharing based solution that is suitable for RFID-enabled supply chains.

However, the JPP mechanism is not secure in that a tag always sends a constant reply to any query. As a result, the tag is vulnerable to tracking by

either supply chain parties or outsiders. Also, an adversary can impersonate a tag in replay attack. When the tagged items are handed over from upstream parties to downstream parties in a supply chain, the downstream parties need to adjust the threshold in key recovery as they reassemble the collection/batch of tagged items. The JPP mechanism does not provide such flexibility.

In this paper, we propose a secret update protocol to address the problems of the JPP mechanism. Our secret update protocol enhances the security level of the JPP mechanism against tracking and impersonation attacks. It also makes it convenient for a supply chain party to adjust the threshold in key recovery according to the size of each batch of tags to be processed. Our work enhances the security and practicality of the JPP mechanism with reasonable cost paid to increase the tag's functionality.

The rest of this paper is organized as follows. In Section 2, we review the characteristics of supply chain, the secret sharing approaches, and the JPP mechanism. In Section 3, we elaborate on our secret update protocol. In Section 4, we formally prove the security of our secret update protocol. At last, we conclude this paper.

2 Reviews

In this section, we briefly introduce the background knowledge about the security requirements for RFID-enabled supply chains. We review several secret sharing approaches and revisit the JPP mechanism.

2.1 Security Requirements for RFID-Enabled Supply Chains

As pointed out in [4], the requirements for any security mechanism in RFID-enabled supply chains should be carefully defined according to the supply chain characteristics, which are summarized below.

- ◇ **None pre-existing trust relationship:** The two adjacent parties in a supply chain may have no previous trust relationship. The current owner of the tagged items may not always know which party will take over part or all of the tagged items before it gets the order from the next party.
- ◇ **Unidirectional downsizing:** Items start off in large collections and progressively get whittled down into smaller aggregates as they make their way from upstream parties to downstream parties.
- ◇ **Compulsory processing orders:** A batch of tags that are processed together by a downstream party must be processed together by an upstream party.

As such, a schematic representing the de- & re-packing of a case containing multiple item tags within a supply chain party is illustrated in Fig. 1 as our running example: *On receiving a case of 100 items transported from an upstream party, the current owner disassembles the case into item-level tags. According to some business orders from downstream parties, the owner then repackages those*

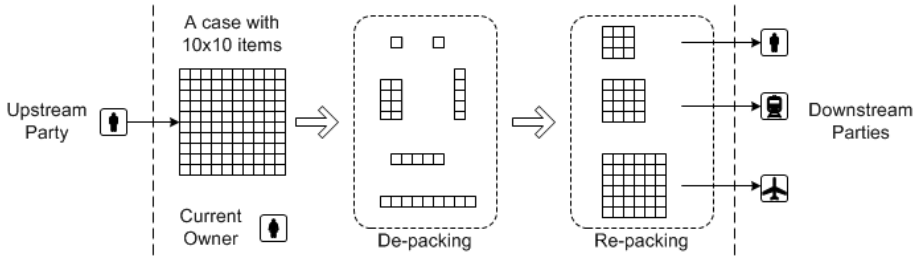


Fig. 1. De- & Re-packing of Item Tags in RFID-enabled Supply Chains. A case of 10×10 items, transported from an upstream party, can be de-packed and re-packed by the current owner. The newly packed cases are to be shipped out to the downstream parties.

items into smaller cases with variable sizes and ships them out. Before shipping out, the owner of the tags would most likely update the tags so that the smaller batches can be efficiently authenticated by the downstream parties.

2.2 Secret Sharing Approaches

Till now, three secret sharing mechanisms [8,9,4] were proposed to solve the key distribution problem in RFID-enabled supply chains.

The Shamir tag proposed by Langheinrich and Marti [8] is the first proposal in this direction. This solution splits the *true ID* of a tag using Shamir's secret sharing scheme [14] and stores all the shares on the tag itself. All the shares form the *new ID* of a tag. Upon a reader's inquiry, an initial set of random bits from the new ID is released, following by subsequent throttled single-bit releases. Eventually, all bits of the new ID will be released and only then can the true ID be computed [8]. The security of this proposal is relevant to the time that an adversary can access the tag. An RFID reader needs a sufficient period of time to collect all bits of a tag's new ID before it can recover the tag's true ID. The time period is typically several minutes for reasonable security. The security of this solution depends on the assumption that a "hit-and-run" adversary cannot access a tag in a long enough time period to collect all bits of a tag's new ID. The problem is, in a supply chain, typically a large number of tags need to be processed in an efficient manner. It may not be practical to spend several minutes to identify a tag. Although the initial set of random bits can be used for fast identifying, it works only when the reader knows all tags' new IDs and true IDs. Since a database of tags is not distributed among supply chain parties (for solving the key distribution problem), a new owner of the tags has no knowledge about the tags' new IDs. Therefore, the Shamir tag scheme is not practical for ownership transfer in RFID-enabled supply chains.

Langheinrich and Marti have also extended Shamir's scheme to distribute an item's ID over hundreds of tags that are attached to or integrated into the item's material [9]. This method may be suitable for protecting containers or

large items; however, it is not practical to authenticate hundreds of tags for each item in supply chains.

In USenix Security 08, Juels, Pappu and Parno proposed a key sharing mechanism (i.e., the JPP mechanism) [4] to enhance the practicality of the two early secret sharing based solutions proposed by Langheinrich and Marti [8,9] by removing the constraints on the period of each tag being read and the number of tags attached to each item. The JPP mechanism is particularly efficient for ownership transfer in RFID-enabled supply chains since it eliminates the need for distributing a database of tag keys among supply chain parties.

2.3 The JPP Mechanism

In the JPP-mechanism, a batch of tags share the same key k , which is split to n shares using a (τ, n) threshold secret sharing (TSS) scheme [14], where $\tau < n$ is a threshold. Anyone who collects at least τ shares can recover k . The JPP mechanism provides two methods to implement the (τ, n) -TSS scheme.

The first one is called “secret sharing across space” which uses Error Correcting Code (ECC) to construct the (τ, n) -TSS scheme. Each tag T_i stores a share s_i as well as its symmetrically encrypted information $E_k(m_i)$, where m_i is the information about the tag T_i . The tag T_i sends $(s_i, E_k(m_i))$ to any reader who queries it.

The other method is called “secret sharing across time”, which uses the “Sliding-Window Information Secret Sharing” (SWISS) scheme to generate the shares of a tag. Each tag stores values of (s_{i1}, s_{i2}, r_i) , where s_{i1} and s_{i2} are two key shares, and r_i is a tag-specific random number. One of key shares is used to derive the common key k of the tags in the timing window to which tag T_i belongs (the timing window determines which key share is used in deriving the key). From k and r_i , a reader can derive an individual key¹ k_i for each tag T_i , which is used to symmetrically encrypt the tag information m_i as $E_{k_i}(m_i)$.

We can see that the JPP mechanism is based on spreading a common secret into different tags. To summarize, we denote the tag’s content as (S_i, M_i) , where S_i represents the values used to derive the common key k , and M_i is the information related to the tag T_i itself. In the “secret sharing across space” method, we have $S_i = s_i$ and $M_i = E_k(m_i)$, while in the “secret sharing across time” method, we have $S_i = s_{i1} || s_{i2}$ and $M_i = r_i || E_{k_i}(m_i)$, where k_i is derived from the common key k .

The JPP mechanism is secure under the assumption that an adversary cannot get access to enough shares for recovering a tag key in the “open area” (e.g., retail stores or customer homes), while legitimate supply chain parties can collect enough shares for recovering each tag key in the “closed area” of a supply chain, to which the adversary does not have access.

¹ Note that although in the “secret sharing across time” method, each tag has a separate key k_i , the key k_i is derived from a common key k and r_i as $k_i = h(r_i, k)$, where h is a hash function. Since r_i is available to any reader, this method does not provide any stronger security than the “secret sharing across space” method in which all the tags shares a common key.

3 Secret Updating for Unidirectional Key Distribution

3.1 Observations

One of the highlights of the JPP mechanism is that it is suitable for EPC-global Class-1 Generation-2 tags. The reason is that JPP mechanism requires no processing power on the tags and that the storage of EPCglobal Class-1 Generation-2 tags is enough to hold the tag information. This means that the JPP mechanism can easily be deployed in current RFID systems without changing the requirements to the tags. It also eliminates the need of distributing a database of tags among supply chain parties for efficient ownership transfer in RFID-enabled supply chains. However, we have two key observations over the JPP mechanism.

Firstly, the adversary model of the JPP mechanism is not too strong in that the tags are assumed to be secure within the “closed area” of a supply chain, while the tags are exposed to an adversary only in the “open area.” In practice, the security of supply chain is arguable; the parties in a dynamic or ad hoc supply chain may not trust or even know each other before they transfer tags to each other. It is more reasonable to assume that the adversary’s power is limited rather than the tags are secure. Under this assumption, we have the following security observation on the JPP mechanism.

Vulnerable to tracking: In the JPP mechanism, a tag T_i always sends the same reply (S_i, M_i) to any reader who queries it. Although an adversary may not get enough shares to decrypt the content of the tag, the never-changing reply can be used by the adversary to track the tag.

Vulnerable to counterfeiting: As the public accessible message (S_i, M_i) is used for a reader to identify the tag T_i , an adversary can easily fabricate a tag that also sends (S_i, M_i) , and replace the tagged item with the fabricated tag.

Secondly, the realistic deployment of the JPP mechanism is largely restricted by the so called monopolistic key assignment model, in which a monopoly (typically the manufacturer of the goods) pre-assigns all the keys (shares) to the tags according a fixed secret sharing scheme with conjectured parameters. Under this assumption, the monopoly has to know or predict much detailed information on how goods are de-packed and re-packed by the downstream parties along a supply chain. If the threshold of the secret sharing scheme is set to be too large, it is possible that the batch size becomes smaller than the threshold so that a valid downstream party cannot promptly collect enough shares to recover the key k . If the threshold is set to be too small (so that all supply chain parties can easily collect enough shares), it may also be easier for an adversary to collect enough shares, especially in upstream supply chains. Without the knowledge that is either not known, or hard to predict, and subject to uninformed changes, a proper threshold is hard, if not impossible, to be decided. Thus, this *one-size-fits-all* solution is not flexibly applicable.

In the unidirectional supply chain channel, no one has better knowledge on how to de-pack and re-pack the tagged goods than the current owner. As in our running example shown in Fig. 1, a 10×10 case, in which all the tags can be originally assigned with shares by a $(33, 100)$ -TSS scheme, is de-packed and re-packed into smaller cases by the current owner such that the tags in a 3×3 case can be re-assigned with secret shares by a $(3, 9)$ -TSS scheme, and the tags in a 6×6 case with shares by a $(12, 36)$ -TSS scheme. It is highly demanded that this current owner be able to update the tags according to the status quo. We thus promote a flexible unidirectional key distribution scheme in RFID-enabled supply chains, where an intermediate supply chain party can flexibly and securely update the secrets (or shares) on the tags that are currently in processing, with some secret update protocol as proposed below.

3.2 The Secret Update Protocol

In order to solve the security and flexibility problems of the JPP mechanism while maintaining its merits, we propose a flexible and secure secret update protocol as an enhancement of the JPP mechanism. The purpose is to change the tag message (S_i, M_i) by each supply chain party to prevent the tracking and counterfeiting across supply chain parties; the secret update protocol may also change the parameters of the secret sharing scheme to adapt to the changes in collection size in supply chains.

A tag must verify the validity of the reader when executing the secret update protocol, or else any malicious reader can rewrite the tag. In the JPP mechanism, the only difference between the valid reader and unauthorized reader is the valid reader can recover the key k . Consequently, a tag can verify the reader's validity by checking the reader's possession of k .

We realize the reader authentication by adding a value $c_i = h(k||S_i)$ on the tag, where $h : \{0, 1\}^* \rightarrow \{0, 1\}^l$ is a one-way hash function, and l is the security parameter of the system². In our scheme, c_i is used by the tag T_i to authenticate the reader. The secret update protocol updates all the values stored on the tag, namely (S_i, M_i, c_i) , to a set of new values (S'_i, M'_i, c'_i) . The requirement on tags is that the tags should have the ability to perform hash functions and have slightly more storage than the JPP mechanism to store c_i . This is the price to pay for the enhanced security and functionality.

Before the secret update protocol is launched by a reader, the reader is assumed to have recovered the key k with enough shares. It is also assumed that the reader has chosen a new key k' and split it using a pre-chosen secret sharing scheme with new parameters (τ', n') , where typically $\tau' \leq \tau$ and $n' \leq n$ for tags moving towards downstream supply chain. The secret update protocol is shown in Fig. 2 and described in the following.

1. **Reader** \longleftrightarrow **Tag**: The reader first identifies the tag T_i based on the recovered key k and the shared secret c_i , with any privacy-enhanced

² The length of $(S'_i||M'_i)$ is supposed to be 96 bits in [4], then we set l to be 96 bits. Refer to Section 3.5 for more discussions on the implementation issues.

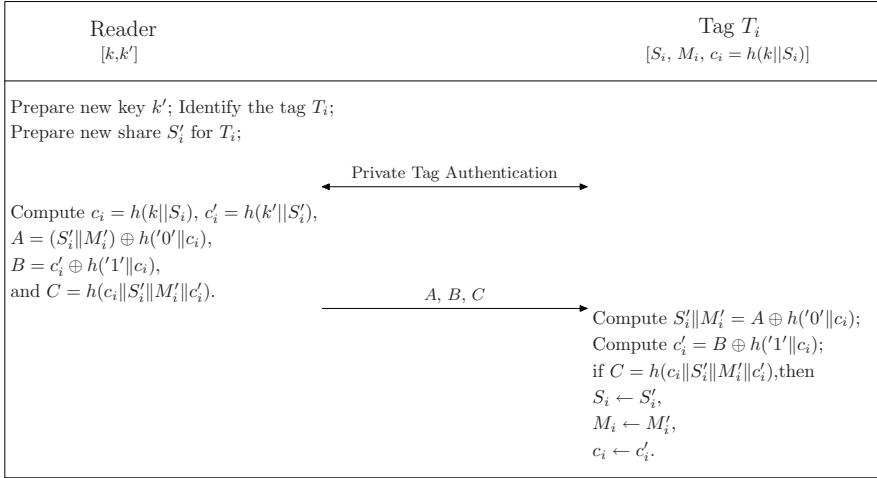


Fig. 2. The Secret Update Protocol

authentication protocol³. Once the tag is authenticated, the reader is ready to prepare the new secret share for the tag's updating.

2. **Reader** \longrightarrow **Tag**: The reader then computes $c_i = h(k||S_i)$, and assigns an unused share S'_i (of the new key k') to T_i . The reader calculates $c'_i = h(k'||S'_i)$, $A = (S'_i||M'_i) \oplus h('0'||c_i)$, $B = c'_i \oplus h('1'||c_i)$, $C = h(c_i||S'_i||M'_i||c'_i)$. Finally, the reader sends (A, B, C) to T_i .
3. **Tag**: After receiving (A, B, C) from the reader, T_i computes $S'_i||M'_i = A \oplus h('0'||c_i)$, $c'_i = B \oplus h('1'||c_i)$. If $C = h(c_i||S'_i||M'_i||c'_i)$, the reader is authenticated. Then T_i updates its values to be $S_i = S'_i$, $M_i = M'_i$ and $c_i = c'_i$.

To make sure that a tag is updated successfully, the reader can identify the tag after the updating process. Recall that in the JPP mechanism, a tag T_i always sends the same reply (S_i, M_i) to the reader who queries it. The reader does not authenticate the tag; thus, an adversary can easily forge a tag by replaying the reply message. In comparison, our secret update protocol restricts this counterfeiting problem from propagating to different supply chain parties, it can be further strengthened to solve the problem even within a supply chain party's territory. To achieve this goal, the first step of the secret update protocol can be any privacy-enhanced tag authentication protocol. For instance, a typical challenge-response tag authentication protocol could be as follows: First, a reader sends its query together with a fresh random number r_1 to a tag. Second, the tag

³ To propose any new private tag authentication protocol is not our major intention in this paper. As an example, a typical challenge-response protocol is described in the following paragraph. Note that if performance is the primary goal, a tag could be quickly identified without being privately authenticated by solely presenting its identifier.

T_i generates a fresh random number r_2 and replies (r_2, D) to the reader, where $D = h(r_1 \| r_2 \| c_i)$. The reader verifies the value of D based on its knowledge on c_i of T_i . Below we discuss the security, performance and implementation aspects of our protocol.

3.3 Security Properties

We list several primary security goals which are desired in a secret update protocol, and discuss how they are achieved in our protocol.

Authoritative access to tags. The security of the secret update protocol relies on the confidentiality of c_i . Given an update message (A, B, C) , only the one who knows the value of c_i can obtain the values of S'_i , M'_i and c'_i . Without knowing c_i , an adversary cannot forge a valid message (A', B', C') due to the security property of the hash function. Thus, unauthorized readers cannot update the tags' secret.

Authenticity of tags. In our updating protocol, the tag T_i is authenticated with any privacy-enhanced authentication scheme (*E.g.*, the challenge-response protocol, where D_i , the authentication message, can be verified only with the knowledge of c_i). The JPP mechanism does not provide tag authenticity since the tag always sends static reply to any query. Thus, our secret update protocol solves the counterfeiting problem.

Forward security. In our secret update protocol, the tag T_i is updated with new values (S'_i, M'_i, c'_i) , which are totally independent from its previous values (S_i, M_i, c_i) . Therefore, the protocol achieves forward secrecy for the tags' content in previous periods.

Untraceability. The JPP mechanism cannot resist tracking attacks, as the tag always sends static values S_i and M_i to any reader who queries it. Our secret update protocol provides the anti-tracking property by updating a tag's secret to a new value, thus an attacker is not able to link the values before and after the updating operation to the same tag (with the assumption that the updating messages are secure against eavesdropping, while an adversary is allowed to track a tag before and after the updating protocol). Note that even if an adversary eavesdrops the updating messages, as long as s/he cannot correlate the messages before and after the updating, s/he still cannot track a tag in different periods. However, if an active adversary is able to monitor the update process and query a tag immediately before and after its updating, s/he can track the tag by correlating S_i and S'_i . Note that such correlation is difficult to achieve in practice because the limited radio frequency range makes the protocol operate in a relatively secure environment, especially when update is performed.

Besides the enhancement of the security level, our secret update protocol solves the reassembly issue since the current owner of a tag can write new values (S'_i, M'_i) into the tag according to new secret sharing parameters (τ', n') . Thus, a downstream party may flexibly choose $\tau' < \tau$ and $n' < n$ for the convenience of

processing smaller batches of tags required in dynamic supply chain environment. The formal proof on the privacy of the tags can be found in Section 4.

3.4 Comparisons

Above we mentioned that secret sharing approaches present a new research direction on solving the key distribution problem in RFID-enabled supply chains. Their advantage can be demonstrated by comparing with several existing RFID authentication protocols that are selected to be classical and representative under a coarse classification. We list the major security and performance metrics of these schemes and compare our solution with them in Table 1.

Table 1. Security and Performance Comparisons

	Requirement for Key Distribution	Resistance to Tracking	Resistance to Counterfeiting	Complexity in Online Identification
Hash chain [12]	yes	yes	no	$O(\log n)$
Hash tree [11]	yes	yes	yes	$O(\log n)$
Pseudonym [10]	yes	yes	yes	$O(n)$
JPP mechanism [4]	no	no	no	$O(1)$
Our protocol	no	yes	yes	$O(1)$

From Table 1, the advantage of using secret sharing based mechanisms is very clear, as they simply solve the key distribution problem which is considered as a cornerstone of cryptography, and is also fraught with complexity in real world applications, while traditional schemes [12,11,10] presuppose the existence of shared keys between mutually trusted parties.

The hash chain-based scheme [12] is efficient since the online identification complexity is $O(\log n)$; however, it cannot resist counterfeiting attack. In the tree-based scheme [11], each tag stores $\log n$ secrets, in which $\log n - 1$ secrets are shared with other tags. It increases the storage requirement on the tags and communication rounds between the tag and the reader, although it also decreases the online identification time to $\log n$. The randomized pseudonym-based protocol [10] assigns each tag with a pseudonymous ID, which is mapped to a real ID stored in the backend database. The protocol can resist tracking and counterfeiting attacks, but its complexity of searching a tag is $O(n)$.

The JPP mechanism is the first applicable solution that is suitable for supply chains and without the need for pre-sharing of the secrets. However, its security level is not sufficient as mentioned above. Our solution enhances the security of the JPP mechanism with additional computational requirements that are comparable with existing solutions, which typically require random number generator and hash function on the tag. In our protocol, the reader can obtain all tags' information after getting τ shares to recover the secret k . The reader also needs to conduct a private authentication protocol for identifying a tag. Besides, each tag needs to store three values which are constant on storage.

3.5 Implementation Considerations

In [4], Juels, Pappu, and Parno implemented a $(15, 20)$ threshold secret sharing scheme on Gen2 tags. In their case, for totally 20 available tags, a reader needs to collect at least 15 tags' shares to successfully recover the secret key and decrypt the encrypted information. The implementation only employs the 96-bit EPC memory bank of a "Alien Squiggle" Gen2 tag, of which 16 bits are used for storing a single share and 80 bits are used for storing the encrypted identity. Note that the shares are generated and written into the tags by encoding a secret key into 20 16-bit symbols using a $(20, 15)$ Reed-Solomon Error Correcting Code (RS-ECC) [13].

Given this parameterization (*w.r.t.*, $|S_i| = 16$ and $|M_i| = 80$), our protocol, replacing (S_i, M_i) with (S'_i, M'_i, c'_i) , requires additional memory space for storing c'_i message (which is equivalent to 160 bits, if SHA-1 cryptographic hash function is used.). As such, we can not put them all into the EPC memory bank, but put (S'_i, M'_i) into the EPC memory bank as in [4], and put (c'_i) into the "User" memory bank [4]. Additionally, we comment that a typical EPC is exactly 96 bits, by applying a block cipher in any authenticated and encrypted method, the encrypted EPC message still has 96-bit length. Hence, a share value (S_i) in [4] might not be properly stored in the EPC memory bank (although its length is only 16-bit), but inevitably be stored in the user memory bank. Fortunately, unlike the memory banks for storing the *Tag unique Identifier* (or "TID") and the *Access* and *Kill* passwords, both the EPC memory and the user memory have similar physical and deployment characteristics (regarding the *password-based lock*, *unlock*, *permalock*, and *password-based write* operations on these memory banks) according to EPCglobal UHF C1 G2 standard [2]. While the JPP mechanism requires only *write-once* EPC memory, our protocol requires *rewritable* EPC memory and user memory on a Gen2 tag. Such a *(re)write* operation is typically allowed in a **secured** state on interrogating a Gen2 tag, which is transitioned from an **open** state by providing the correct *Access* password. On implementing our protocol, we indicate that the 32-bit *Access* password of a Gen2 tag be individually derived from the recovered shared key k (e.g., we obtain the 32-bit *Access* password by hashing the key k concatenated with the tag's EPC code, and then taking the first 32 bits of the output.).

Moreover, in real-world usage, one has to determine the number of tags n processed in a batch and the threshold τ to recover the secret key. Now suppose the current owner in Fig. 1 receives a case of 100 tags, which are formatted with $(\tau, 100)$ -TSS scheme. On choosing a proper threshold, τ can be set as the biggest value (e.g., $\tau = 80$) that is less than certain upper bound to maximally tolerate (up to 20) reading or erasure errors; alternatively, τ can be set as the smallest value (e.g., $\tau = 20$) that is greater than certain lower bound to guarantee the robustness on recovering the key with a minimum number (20) of tags. As such, the owner can recover the secret key and then decrypt the encrypted information

⁴ Note that different tag manufactures will produce different form-factor Gen2 tags all conforming to EPCglobal C1 G2 specification [2]. E.g., a Philips UCODE EPC Gen2 tag contains 224 bits in the user memory bank.

attached with each tag. According to downstream parties' requirements, the owner can de-pack the case and re-pack it into several smaller cases including 4 cases with 4×4 tags and 1 cases with 6×6 tags. Similarly, a flexible ($4 \sim 12, 16$)-TSS scheme and a flexible ($8 \sim 28, 36$)-TSS scheme can be chosen for these two kinds of cases respectively, to either maintain robust recovery with minimal 20% available tags, or tolerate about 20% of reading/erasure errors.

4 Security Proof

In this section, we formally prove the security of our secret update protocol. At first, we formalize the security requirement for the secret update protocol. Then, we show that the proposed protocol satisfies the privacy requirement.

4.1 Security Model

We focus on the privacy property of our secret update protocol. As for the JPP authentication protocol running between an RFID reader and a batch of tags, it has been proven that in [4], it is infeasible for any polynomial time adversary to learn the shared key of the tags if the adversary cannot get access to the replies from at least τ tags. This privacy property is strengthened in our secret update protocol since the shared key of the tags can be updated by each valid supply chain party.

Informally, in the secret update protocol, there are two basic security requirements: (i) The former owner should not be able to trace the current owner. (ii) Any adversary outside the supply chain should not be able to link any two protocol messages in different *periods* (a period is the lifetime of a tag between its being updated by two adjacent owners). A formal privacy definition is described in the following privacy game of secret update protocol (Game PoT for short, which is illustrated in Fig. 3).

We first give a formal description of an RFID system. In an RFID system, there are a set of tags $\mathcal{T} = \{T_1, \dots, T_\ell\}$, a set of readers $\mathcal{R} = \{R_1, \dots, R_m\}$ and an adversary \mathcal{A} . Each tag stores a secret which is updated when its owner has changed. The RFID system is initialized and updated through a (τ, n) secret sharing scheme [14]. To identify a tag, a reader interacts with the tag through the authentication protocol $\pi(R_i, T_j)$. When a tag is passed from one owner to another owner, its current owner R_i runs the secret update protocol $\kappa(R_i, T_j)$ to reset the internal state of tag T_j .

As above, the Game PoT consists of three phases. In the setup phase, the game initializes the RFID system. Then in the learning phase, the adversary \mathcal{A} performs a series of queries to enlarge its knowledge base about the RFID system. In the third phase, the adversary \mathcal{A} chooses two tags for challenging. Then, a tag is chosen by randomly updating one of the two tags. After this, the updated tag is given to the adversary as a challenging tag for him to distinguish it from the original two tags.

In the Game PoT, the adversary \mathcal{A} is allowed to eavesdrop the protocol messages (by invoking an authentication protocol $\pi(R_i, T_j)$ between a reader R_i and

Game PoT	(Game PoT)
Setup:	
(1) Initialize a set of tags $\mathcal{T} = \{T_1, \dots, T_\ell\}$.	
(2) Initialize a set of readers $\mathcal{R} = \{R_1, \dots, R_m\}$.	
Learning phase:	
(3) For any $R_i \in \mathcal{R}$ and $T_j \in \mathcal{T}$, \mathcal{A} may do the following in any order as long as $r_{k,pd} + v_{k,pd} \leq \tau$, where $r_{k,pd}$ and $v_{k,pd}$ denote the number of authentication protocol calls related to tags whose data was encrypted by k in the <i>period</i> pd and the number of corruption calls related to tags whose data was encrypted by k in the <i>period</i> pd respectively:	
(3.1) Make $\pi(R_i, T_j)$ calls, without exceeding r overall calls.	
(3.2) Make $\kappa(R_i, T_j)$ calls, without exceeding u overall calls.	
(3.3) Corrupt tags, without exceeding v overall calls.	
(3.4) Corrupt any reader except the reader R' .	
Challenge phase:	
(4) \mathcal{A} selects two tags T_p and T_q .	
(5) Let $T_0 = T_p$, $T_1 = T_q$ and $b \in_R \{0, 1\}$.	
(6) the game runs $\kappa(R', T_b)$ and then sends T_b to \mathcal{A} .	
(7) For any $R_i \in \mathcal{R}$ and $T_j \in \mathcal{T}$, \mathcal{A} may do the following in any order as long as $r_{k,pd} + v_{k,pd} \leq \tau$:	
(7.1) Make $\pi(R_i, T_j)$ calls, without exceeding r overall calls.	
(7.2) Make $\kappa(R_i, T_j)$ calls, without exceeding u overall calls.	
(7.3) Corrupt tags, without exceeding v overall calls.	
(7.4) Corrupt any reader except the reader R' .	
(8) \mathcal{A} outputs a guess bit b' .	
(9) If $b' = b$ then output 1, else 0.	

Fig. 3. Privacy Game of Secret Update Protocol

a tag T_j) and to corrupt the tags under the restriction that the sum of the number of calls of the authentication protocol and the number of tag corruptions is at most τ (i.e., the threshold of the secret sharing scheme) during the same *period*. The adversary \mathcal{A} is also allowed to invoke secret update protocol $\kappa(R_i, T_j)$ with restriction that it cannot learn the messages in the secret update protocol. Note that this assumption is commonly used in most secret update protocols and it is reasonable since the secret update protocol is usually executed in a relative secure environment such as the warehouse of a supply chain party. The adversary \mathcal{A} is allowed to corrupt any readers (and hence to get their internal states) except the reader R' to which the challenging tag is presented.

The goal of the adversary \mathcal{A} in the Game PoT is to distinguish between two different tags chosen by itself. The adversary may know the internal state of both tags. But the challenging tag is chosen by updating a tag selected randomly from the two tags.

Definition 1. A function $f : N \rightarrow R$ is said to be negligible if for every $c > 0$ there exists a number $m \in N$ such that $f(n) < \frac{1}{n^c}$ holds for all $n > m$. Also, a function $f : N \rightarrow R$ is said to be overwhelming if $1 - f(n)$ is negligible.

Definition 2. The advantage of adversary \mathcal{A} in the Game PoT is defined as

$$\text{Adv}_{\mathcal{A}}(r, u, v, \tau, n, \ell, m) = |2\text{Pr}[\text{Game PoT outputs 1}] - 1|.$$

Definition 3. We call that the RFID system is $(r, u, v, \tau, n, \ell, m, t, \epsilon)$ -private, if there exists no polynomial probabilistic time adversary \mathcal{A} whose advantage $\text{Adv}_{\mathcal{A}}(r, u, v, \tau, n, \ell, m)$ is at least ϵ and whose running time is at most t in the Game PoT.

4.2 Privacy of the Secret Update Protocol

The privacy of our secret update protocol relies on the privacy of the underlying secret sharing scheme. We now prove that our secret update protocol is $(r, u, v, \tau, n, t, \epsilon)$ -private if the underlying secret sharing scheme is $(\tau, n, t, \epsilon/2)$ -private. The following theorem characterizes the security of our secret update protocol⁵.

Theorem 1. In the random oracle model, if the secret sharing scheme used in our RFID protocol is $(\tau, n, t, \epsilon/2)$ -private, then the proposed RFID system is $(r, u, v, \tau, n, \ell, m, t, \epsilon)$ -private.

5 Conclusion

In this paper, we design a flexible and secure secret update protocol as an extension to Juels, Pappu and Parno's unidirectional key distribution scheme [4]. Our secret update protocol provides desirable flexibility so that downstream supply chain parties can adjust the threshold in key recovery for the convenience of processing smaller batches of tags. Moreover, the secret update protocol makes it particularly difficult for the tags to be tracked across multiple supply chain parties. Compared with previous works, our solution is similar in that it does not require a database storing keys of tags being distributed to different supply chain parties, while it is more secure against tracking and counterfeiting attacks, and is more flexible in addressing the reassembly problem. Although our secret update protocol requires more powerful tags than the EPCglobal Class-1 Gen-2 tags, it is worth to pay for the enhanced security and functionality. Our future work will focus on the minimization of the cost of tags in our design.

Acknowledgment. This work is partly supported by A*Star SERC Grant No. 082 101 0022 in Singapore.

References

1. Cai, S., Li, T., Ma, C., Li, Y., Deng, R.: Enabling secure secret updating for unidirectional key distribution in rfid-enabled supply chains, <http://icsd.i2r.a-star.edu.sg/staff/tieyan/SecureRFID/docs/ICICS09-full.pdf>
2. EPCglobal. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz-960 MHz, version 1.2.0. (October 2008)

⁵ The privacy proofs of the secret sharing protocol and the secret sharing scheme are eliminated here due to page limit, but are provided in the full version [1].

3. Fishkin, K., Roy, S., Jiang, B.: Some Methods for Privacy in RFID Communication. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 42–53. Springer, Heidelberg (2005)
4. Juels, A., Pappu, R., Parno, B.: Unidirectional key distribution across time and space with applications to rfid security. In: 17th USENIX Security Symposium, pp. 75–90 (2008)
5. Juels, A.: RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications* 24(2), 381–394 (2006)
6. Juels, A., Rivest, R., Szydlo, M.: The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. In: Conference on Computer and Communications Security – ACM CCS 2003, October 2003, pp. 103–111 (2003)
7. Karjoth, G., Moskowitz, P.: Disabling RFID Tags with Visible Confirmation: Clipped Tags Are Silenced. In: Workshop on Privacy in the Electronic Society – WPES 2005 (November 2005)
8. Langheinrich, M., Marti, R.: Practical Minimalist Cryptography for RFID Privacy. *IEEE Systems Journal, Special Issue on RFID Technology* 1(2), 115–128 (2007)
9. Langheinrich, M., Marti, R.: Rfid privacy using spatially distributed shared secrets. In: Ichikawa, H., Cho, W.-D., Satoh, I., Youn, H.Y. (eds.) UCS 2007. LNCS, vol. 4836, pp. 1–16. Springer, Heidelberg (2007)
10. Li, Y., Ding, X.: Protecting RFID Communications in Supply Chains. In: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security ASIACCS 2007, Singapore, pp. 234–241 (2007)
11. Molnar, D., Wagner, D.: Privacy and Security in Library RFID: Issues, Practices, and Architectures. In: Conference on Computer and Communications Security – ACM CCS 2004, October 2004, pp. 210–219 (2004)
12. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient Hash-Chain Based RFID Privacy Protection Scheme. In: International Conference on Ubiquitous Computing – Ubi-comp 2004 (September 2004)
13. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. *Journal SIAM* 8, 300–304 (1960)
14. Shamir, A.: How to share a secret. *Commun. ACM* 22(11), 612–613 (1979)
15. Weis, S., Sarma, S., Rivest, R., Engels, D.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) Security in Pervasive Computing. LNCS, vol. 2802, pp. 454–469. Springer, Heidelberg (2004)

Biometric-Based Non-transferable Anonymous Credentials^{*}

Marina Blanton¹ and William M.P. Hudelson²

¹ Department of Computer Science and Engineering, University of Notre Dame
mblanton@cse.nd.edu

² Mathematics Department, Pennsylvania State University
phil.hudelson@gmail.com

Abstract. This work explores the problem of using biometric data to achieve non-transferability of anonymous credentials; that is, sharing of anonymous credentials, which allow one to anonymously authenticate, can be severely limited if their use requires possession of the credential owner's biometric. We target to provide strong security guarantees using minimal trust assumptions, namely that a fresh reading of a biometric is enforced on each use of the credentials. Furthermore, no biometric or other information is compromised if an adversary obtains full access to all credential-related data. Our solution relies on constructions for fuzzy extractors that allow one to extract and reproduce a random string from noisy biometric images. We first examine security requirements of biometric key generators, and then show how they can be integrated with anonymous credentials to achieve a high degree of non-transferability and security.

1 Introduction

Biometric-based authentication is becoming more prevalent today. This is well justified by the advances in biometric recognition techniques and a higher degree of security of biometric-based authentication compared to some alternative authentication mechanisms. Biometric-based authentication is also viewed as convenient to users, as it does not require them to remember passwords or carry authentication tokens or keys. Biometric data, however, requires very careful handling and protection, since, once captured, it can uniquely identify an individual and cannot be revoked. For that reason, a lot of research in the recent years has been dedicated to designing mechanisms that minimize the impact of (accidental or intentional) leakage of biometric data stored in databases for authentication purposes. Instead of working on protecting privacy of biometric data, the direction explored in this work comes from the (opposite) idea of using biometrics to aid privacy and anonymity. In particular, biometric data can be incorporated into privacy-preserving tools to limit abuse of anonymity, and thus enable a safer deployment of such techniques on a wider scale.

Anonymous authentication allows the prover to convince the verifier that she has a certain property or credential without revealing any other information (and without

^{*} Portions of this work were sponsored by grant AFOSR-FA9550-09-1-0223. This work was performed while the second author was at the University of Notre Dame.

the ability of two authentication instances to be linked to the same individual by the verifier). In the case of theft or voluntary sharing of such credentials by duplication, these credentials can be freely used by others without the ability to find out the identity of their owner. Incorporating biometrics in the authentication process then results in a high degree of assurance that credentials cannot be transferred and that access is performed by credential bearers only. We arrive at a setting where biometrics is used to ensure non-transferability of anonymous credentials realizing the concept of accountable anonymity. The benefits of such an approach are clearly seen in numerous applications where non-transferability is essential. For example, in an organization where digital locks are used to enter buildings and facilities, it is extremely easy to track one's movements, and the employer that respects individual privacy can be willing to implement anonymous access to its facilities, i.e., the same as conventional keys provide¹. Digital keys can then be issued in the form of anonymous credentials that encode access privileges of an employee, and should not be used by any other individual. Other important applications of non-transferable anonymous credentials include certification that the credential-bearer is a U.S. citizen, guaranteeing that the insurance company will pay for an anonymous HIV test of the credential-bearer, and others.

When biometry is integrated into the authentication process, to achieve unlinkability of protocol executions, biometrics can no longer be scanned by the device that performs access control enforcement. This means that biometrics are captured on a device that each user carries and which is trusted by the system. Then during the enrollment phase, user credentials are placed on that device, and during authentication the device is trusted to capture a biometric and use the scan in combination with the user credentials to authenticate anonymously. While anonymous credentials schemes where biometrics is used to achieve non-transferability have been proposed in the past [5,24], the key difference between our and prior work is in the trust requirements we place on the tamper-resistant device. In our case these requirements are minimal: the only functionality the device is required to do correctly is to enforce capturing a new biometric (and erase it afterwards). Should the integrity of the device be compromised and all information stored on it retrieved, it is not feasible to either recover the biometric data that identifies the credential-bearer or to successfully authenticate using the captured credentials (this is achieved without reliance on any additional secrets that the credential-bearer must know).

The structure and contributions of this work are as follows: We first discuss biometrics key generators, which allow one to extract cryptographic keys from biometric data, including their common constructions and security requirements. We then propose a generic way of improving security of biometric key generators with respect to privacy protection of the biometric from which such keys are derived. In addition to being useful for biometric key generators, this construction has a direct application to our anonymous credentials scheme where biometrics are used to achieve non-transferability. The next part of this work is dedicated to constructing a general solution for such anonymous credentials, to which we refer as biometric-based non-transferable credentials.

¹ Note that anonymous access will not be suitable for certain organizations where access to information or restricted-access facilities must be monitored by law or other provisions, but in most environments anonymous access to facilities is natural and harmless if access control is properly implemented.

The only requirement that we pose to ensure correct operation of the system is that biometric readings are enforced on each use of the credentials (and erased afterwards). Once again, this offers protection of credentials in cases of both voluntary duplication of anonymous credentials and when such credentials are stolen.

2 The Model and Preliminaries

2.1 The Model

The operation of our system proceeds in a standard way: First, an authority A sets up the system, which includes generation of a public-private key pair (pk_A, sk_A) , where the private key will allow it to issue user credentials and the public key can be used to verify them. When a user joins the system, her biometric is recorded and credentials are issued by the authority in accordance with user privileges. During the authentication protocol, the user's biometric is captured by her device and the credentials are verified by the authority (or a different entity on behalf of the authority) in an anonymous way. More precisely, an authentication scheme consists of the following components:

- AC-Setup is an algorithm that, on input a security parameter κ , sets up the system including A 's public-private key pair (pk_A, sk_A) . Its output consists of public parameters pub that include pk_A , and A 's secret key sk_A .
- AC-Enroll is a procedure during which, given system's parameters pub , a user \mathcal{U} 's input consists of her biometric, the authority's input consists of its secret key sk_A and \mathcal{U} 's privileges. It results in \mathcal{U} obtaining credentials cred that are tied to her biometric and specify her access privileges.
- AC-Auth is a protocol between a user \mathcal{U} and server \mathcal{S} , in which \mathcal{U} 's input consists of her fresh biometric reading and credentials cred , and the server's input consists of public parameters pub . Authentication is successful if the server could successfully verify \mathcal{U} 's credentials as authentic and meeting the access control policy.

A secure anonymous biometric-based authentication scheme must satisfy the following:

Completeness: Every honest user should be able to successfully authenticate using her biometric and credentials.

Soundness: A person without proper credentials should not be able to successfully authenticate with more than negligible (in κ) probability after observing any (polynomial) number of successful authentication protocols. Furthermore, any coalition of valid users should not be able to gain access to more resources than what they can already legitimately access with more than negligible probability.

Unlinkability: To satisfy anonymity requirements, we require that one should not be able to determine with more than negligible probability whether two executions of the authentication protocol correspond to the same user or different users.

Privacy of biometric: We require that in the case of compromise of a user's device, the information stored on it does not allow one to learn the credentials owner's biometric or successfully authenticate without the knowledge of it.

These properties are defined formally in section [4.2](#)

2.2 Signatures with Protocols

We use signature schemes due to Camenisch and Lysyanskaya [12,13], which have two protocols associated with them: (i) they allow a user to obtain a signature on a committed value without revealing that value to the signer; and (ii) they enable a user to convince a third party that she possesses a signature on a certain value. The commitment scheme used is the Pedersen commitment [32], in which the public parameters are a group G_q of prime order q such that the discrete logarithm problem is hard in G_q and generators g_0, g_1, \dots, g_ℓ . To compute a commitment to $x_1, \dots, x_\ell \in \mathbb{Z}_q$, we randomly choose $r \in \mathbb{Z}_q$ and set $\text{com}(x_1, \dots, x_\ell; r) = g_0^r \prod_{i=1}^{\ell} g_i^{x_i}$. This commitment is unconditionally hiding (i.e., $\text{com}(x_1, \dots, x_\ell; r)$ reveals no information about x_1, \dots, x_ℓ) and is computationally binding (assuming that the discrete logarithm problem is hard in G_q , the sender cannot open the commitment to values other than x_1, \dots, x_ℓ).

Then given a commitment $\text{com}(x_1, \dots, x_\ell; r)$, it is possible obtain the signer's Camenisch-Lysyanskaya (CL) signature $\sigma(x_1, \dots, x_\ell)$ without revealing any information about the values x_1, \dots, x_ℓ to the signer. Furthermore, possession of $\sigma(x_1, \dots, x_\ell)$ allows its owner to use commitments to x_1, \dots, x_ℓ to prove to other parties that she has the signer's signature on the values included in the commitments without revealing additional information about the signed values themselves. If this protocol is combined with a zero-knowledge proof that the values included in these commitments satisfy certain properties, it becomes possible to convince a third party that the prover possesses a CL signature that meets these conditions without disclosing additional information about the signed values. The signature scheme [12] relies on the Strong RSA assumption for its security. The scheme [13] relies on LRSW assumption in groups with bilinear maps.

2.3 Zero-Knowledge Proofs of Knowledge

Zero-knowledge proofs of knowledge (ZKPKs) allow one one party, the prover, to prove to another, the verifier, the veracity of some statement without revealing to the verifier any information besides the fact that it is valid. Prior literature provides efficient ZKPKs for a variety of statements, with many efficient proofs being based on the discrete logarithm problem (see, e.g., [16,15,14,6,9]). ZKPKs used in our protocols are a proof of knowledge of the discrete logarithm representation, equality of discrete logarithms, and conjunction of two or more statements [15], solutions to which are well known. Verification of context-specific user privileges might include other techniques such as, e.g., a proof of knowledge that a committed values lies in an interval and others.

3 Biometric Key Generators

Before proceeding with our anonymous credentials scheme, we give a brief description of a *biometric key generator* (BKG), which is a system that allows one to produce a cryptographic key from a biometric and later reconstruct the key using the same (noisy) biometric. Constructions for biometric key generators will be directly used in our biometric-based anonymous authentication scheme, and the results presented in this section have uses in both biometric key generation and our construction of non-transferable anonymous credentials. In what follows, we assume that the system is first

initialized using a security parameter κ , which is used to determine other parameters and algorithms' configurations (i.e., BKG-Setup algorithm is implicit).

BKG-Enroll: a probabilistic algorithm that, on input a user \mathcal{U} 's identity and a biometric representation W , outputs a cryptographic key K and *public information* or *helper data* P that will aid key recovery. If the input does not meet certain criteria, the algorithm might output a special failure symbol \perp .

BKG-KeyRec: a deterministic algorithm that, on input a biometric representation W' and helper data P , outputs either a cryptographic key K or a failure symbol \perp if it is unable to compute a key from its input.

Normally, during the enrollment phase, a cryptographic key is chosen anew and locked with the biometric, or is derived from the biometric. The purpose of the helper data is, given a noisy biometric image W' , to correct the errors and permit unlocking or derivation of the key. Recovery of the correct key is possible only if the biometric representations W and W' are close enough, for some definition of distance specific to the type of biometrics and BKG construction used. Helper data P is designed to leak as little information about the biometric as possible, so that it can be assumed to be non-private data.

Recent work of Ballard et al. [3] lists security properties that must hold for a BKG. We define them next with the difference that we additionally require privacy of the biometric to hold when a user \mathcal{U} is enrolled in the system more than once using the same (noisy) biometric and the same or a similar key generation mechanism. In other words, we want biometric information remain protected when \mathcal{U} 's key is lost or compromised and it re-enrolls, when it legitimately assumes more than one role within the system, or when it is enrolled at more than one system that use similar biometric key generators.

- *Key randomness* (REQ-KR): A key K contains sufficient amount of randomness (based on the security parameter κ) and appears random to any adversary with access to the helper data P used in deriving K and any auxiliary information.
- *Weak biometric privacy* (REQ-WBP): Given helper data P and any auxiliary information, any adversary does not learn useful information about the biometric W used in generating P . Often the difficulty of recovering the entire biometric W can be sufficiently well measured, but it is desirable that learning parts of the biometric is also difficult (i.e., we might desire to prevent an adversary from learning any function of the biometric).
- *Strong biometric privacy* (REQ-SBP): Given helper data P , any auxiliary information, and the key K itself, any adversary does not learn any useful information about the biometric W used in generating P . Similar to the above, we might require that an adversary is unable to compute any function of the biometric.

In the above, the auxiliary information refers to any additional information that can surround the system. Such information available to an adversary can contain biometric data, corresponding helper data, and keys, not associated with the user in question. It can also contain information about distributions of biometric data, implementation decisions, and other information from the environment that can potentially weaken the security of the keys or biometrics.

In recent years, a large number of proposals for biometric key generators have been developed. Examples of thoroughly analyzed and well studied constructions include fuzzy vaults [25] (and work that builds on them [17,36,34,37,29,28,27,30]), secure sketches and fuzzy extractors [21,7,26,8,20] (and work that builds on them [23,21,0,1]). Many of them were designed and adapted to work on real data before the security requirements were defined in the current form. For that reason, many existing constructions and implementations would fail to achieve the strong biometric privacy requirement (and some proposals can fail other requirements as shown in [3]). Here we show that property REQ-SBP is not hard to achieve if a BKG construction can meet the REQ-WBP requirement.

Before we present our generic conversion from REQ-WBP to REQ-SBP, we first need to provide more detail about how BKGs normally work. Secure sketches [21,20] were introduced as a mechanism of correcting errors in noisy secrets (e.g., biometrics) by releasing a helper string S that does not reveal a lot of information about the secret. A secure sketch is generated using a “sketch” procedure SS that, given W , produces a string S ; and can be “recovered” using Rec that given W' and S , outputs W if the distance between W and W' , $\text{dist}(W, W')$, in the appropriate metric space is at most t . Secure sketches have been constructed for different types of metric spaces including the Hamming distance (applicable to iris codes, which are represented as binary strings) and set intersection (applicable to fingerprints represented as a set of points in a two-dimensional plane). Security of a secure sketch is evaluated in terms of leakage of biometric information associated with the release of helper data, i.e., the difference between the “worst-case” entropy of W and the average minimum entropy of W after the release of S .² Since secure sketches are normally built using error-correcting codes, the larger the number of errors that need to be corrected, the more redundancy needs to be included in the code, and the less effective it is at protecting information about the biometric stored in the helper data.

Fuzzy extractors allow one to extract randomness from W (for use in cryptographic constructions) and later reproduce it exactly using different W' close to the original W . A fuzzy extractor is generated by Gen that, on input W , outputs extracted random string R and a helper string P ; and can be reproduced by Rep that on input W' and P outputs R that was generated using $Gen(W)$ if $\text{dist}(W, W') \leq t$. The security requirement is such that, for any W with sufficient entropy, R is sufficiently close to a uniformly chosen random string, even after observing the helper data P . A fuzzy extractor can be built from a secure sketch as follows [21]: on input W , Gen executes $S \leftarrow SS(W)$ and applies a strong extractor Ext to W to extract a random string R . S and random coins used by Ext form the helper data P . Let r_1 denote the random coins used by SS and r_2 random coins used by Ext (i.e., execution is of the form $SS(W; r_1)$ and $Ext(W; r_2)$). We obtain $P = (S, r_2)$. Algorithm $Rep(W', P)$ uses S from P to recover the original W (given that $\text{dist}(W, W') \leq t$) and extracts R by computing $Ext(W, r_2)$. Fuzzy vault is a secure sketch construction for the set intersection metric, so we will use secure sketch and fuzzy extractor terminology in the rest of this paper.

One complaint regarding the theoretical work on biometric key generation is that proposed solutions cannot tolerate realistic variations in the biometric signal such as

² We refer the reader to [21] for precise definitions.

variable-length representations, unordered or unaligned representations [35]. In fact, the unlocking algorithm in the fuzzy vault implementation given by Uludag et al. [34] produces many potential keys, one of which must be selected as authentic. Uludag et al. [34] address this issue by adding some structure (i.e., redundancy) to the secret being locked, thus weakening the hiding properties of the construction. Such variations in the biometric, however, are not arbitrary and often key recovery still can be performed using the proposed schemes. Thus, to aid recovery of the exact secret among several candidates, we propose to ship helper data with a verification value that will permit confirmation of the correct key. Adding a verification value computed using a one-way function can cleanly avoid additional information leakage. Now we are ready to describe our construction.

Assume that we are given a BKG $(\text{BKG-Enroll}_0, \text{BKG-KeyRec}_0)$ that achieves REQ-WBP, but BKG-KeyRec_0 might output several key candidates rather than a single key. Let f be a one-way function that hides all information about its inputs and let $\|$ denote string concatenation. We then simultaneously address key confirmation and strong biometric privacy using the following construction:

$\text{BKG-Enroll}(W)$

1. Run $(K_0, P_0) \leftarrow \text{BKG-Enroll}_0(W)$.
2. Set $P = (P_0, f(K_0\|“0”))$ and $K = f(K_0\|“1”)$.
3. Output (K, P) .

$\text{BKG-KeyRec}(W', P = \{P_1, P_2\})$

1. Run $\text{BKG-KeyRec}_0(W', P_1)$ to find a set \mathcal{K} of candidate keys for K_0 .
2. For each $k \in \mathcal{K}$, if $f(k\|“0”) = P_2$, set $k = K_0$ and output $K = f(K_0\|“1”)$.

Theorem 1. *Given a BKG $(\text{BKG-Enroll}_0, \text{BKG-KeyRec}_0)$ that satisfies REQ-WBP and a one-way function f that hides information about its inputs, the above construction for $(\text{BKG-Enroll}, \text{BKG-KeyRec})$ satisfies REQ-SBP.*

Proof. The proof is straightforward. Given $P = \{P_1, P_2\}$, K and auxiliary information aux , we need to show that an adversary \mathcal{A} does not learn useful information about W . By the assumption that $(\text{BKG-Enroll}_0, \text{BKG-KeyRec}_0)$ satisfies REQ-WBP, \mathcal{A} does not learn information about W from P_1 and aux . The only other information available to \mathcal{A} is P_2 and K , which, by our assumption, were produced using a one-way function that hides information about its inputs. This means that \mathcal{A} cannot gain information about K_0 or W from these data. \square

4 Non-transferable Anonymous Credentials via Biometrics

Now we proceed with combining biometric key generators with anonymous credentials to result in anonymous biometric-based authentication with non-transferable credentials. Recall that we want a user’s credentials to encode a cryptographically strong value derived from the user’s biometric. This value is not stored with the credential (and cannot be recovered from the credential) and must be recomputed from the biometric on each use. To ensure that the value was actually derived from the biometric, we want

the derivation procedure to be one-way, and a proof of derivation should be provided at authentication time in zero knowledge.

In our construction of non-transferable anonymous credentials we crucially rely on the following fact that holds true for known secure sketch constructions and allows us to build a simple and efficient solution.

Fact 1. *Let $P \leftarrow \text{SS}(W; r_1)$, where r_1 denotes the random choices the algorithm SS makes. Then knowledge of W' , such that $\text{dist}(W, W') \leq t$, implies knowledge of r_1 and W . Furthermore, knowledge of r_1 implies knowledge of W and therefore some W'' with $\text{dist}(W, W'') \leq t$.*

Since the trust requirement we place on the device is to enforce a fresh biometric reading, it would be necessary to provide assurance that the key was actually derived from W' . Using the above fact, however, we have that knowledge of one of W , r_1 , and W' is equivalent in presence of P . Thus, if an adversary is able to recover r_1 or W from P and then compromise the integrity of the device to avoid enforcing a new biometric scan, it will be able to successfully produce W' and pass the verification procedures of the protocol. The above fact then mitigates the need to provide expensive zero-knowledge proofs of correct computation of W from W' (procedure Rec) and lets us enforce only a proof that the key was derived from the original biometric W (and, as mentioned above, we want the derivation process to be one-way). Fuzzy extractors were not explicitly designed to be one-way and in fact are not guaranteed to be one-way (i.e., the underlying randomness extractors do not have this property). Thus, we enforce the one-way requirement of key derivation using the construction for REQ-SBP given in the previous section. Note that in this case the REQ-SBP property is not as crucial as in the case of BKGs , because with anonymous biometric-based authentication the derived key does not leave the client; instead, one-wayness of the computation ensures the verifier that the necessary value was actually derived from biometric data.

In our construction, we assume that fuzzy extractor parameters are chosen in such a way as to output strings indistinguishable (to a polynomial-time distinguisher) from strings chosen uniformly at random (for the appropriate choice of the security parameter). That is, using the terminology of randomness extractors, we assume that the fuzzy extractor is configured to output strings ϵ -close to uniform, where ϵ is a negligible function of the security parameter. See [31] for more detail. This makes the output suitable for use in cryptographic protocols.

4.1 The Scheme

To be able to prove to the server that the computation was performed as prescribed, the application of one-way function f must be verifiable in zero-knowledge. To achieve this, we use the verifiable random function (VRF) of Dodis and Yampolskiy [22] to implement f . This function is computed as $y = f_{sk}(x) = g^{1/(sk+x)}$ over groups of prime order with bilinear maps, where g is the group generator, sk is the secret key, and x is the input to the function (which can be public); its security is based on q -DHI and q -DBDHI assumptions (see [22] for more information). In particular, we will assume that the verification value and the extracted key are computed as $f_{sk}("0")$ and $f_{sk}("1")$, where sk is derived from W' and the fuzzy extractor helper data (and must meet the requirements for f).

AC-Setup: On input security parameter 1^κ , authority A sets up the group parameters and its public-private key pair (pk_A, sk_A) for the CL signature scheme. All values except the signing key sk_A are stored in pub .

AC-Enroll:

1. Biometric W of user \mathcal{U} is measured with small error, $\text{BKG-Enroll}(W)$ is executed to produce (P, K) , where $P = (P', V)$ such that (P', K') is the output of the fuzzy extractor Gen algorithm and $V = f_{K'}("0")$ denotes the verification data.
2. \mathcal{U} chooses z_1 at random and sends commitment $Z_1 = g^{z_1}$ to A .
3. A verifies the validity of the computation in step 1 and computes $K = f_{K'}("1")$.
4. A chooses random z_2 and uses Z_1 to compute commitment $\text{com}(K, \text{priv}; z)$, where $z = z_1 + z_2$ (i.e., both A and \mathcal{U} contribute randomness, but the value of z will be known only to the user).
5. A uses sk_A to produce signature $\sigma_A(K, \text{priv})$ and sends it and z_2 to \mathcal{U} .
6. \mathcal{U} computes $z = z_1 + z_2$ and verifies the validity of the signature.
7. User \mathcal{U} 's credentials $\text{cred} = (P, \text{priv}, \sigma_A(K, \text{priv}))$ are stored at his device.

Technically speaking, the signature here is a signature on values K , priv , and random value z , and the value of z will be necessary for showing the validity of the signature. Thus, it is implicitly assumed that this value is also stored with the user's credentials. With this setup, authority A learns biometrics and keys of users and is expected to erase such information after the enrollment protocol. This does not permit A to distinguish between different users at authentication time, but it might be desirable to prevent A from learning user biometrics at all; we leave this as a direction for future work.

AC-Auth: A user \mathcal{U} with a device holding credentials $\text{cred} = (P, \text{priv}, \sigma_A(K, \text{priv}))$ engages in interaction with a server \mathcal{S} as follows:

1. The device scans the user's biometric W' , recovers K' using P' and confirms it using V stored in cred . It then forms commitment $C_1 = \text{com}(K'; z_1)$.
2. The device computes $K = f_{K'}("1")$ and a commitment to it $C_2 = \text{com}(K; z_2)$.
3. The device computes commitment $C_3 = \text{com}(\text{priv}; z_3)$ using priv from \mathcal{U} 's credentials.
4. The device sends to \mathcal{S} C_1, C_2, C_3 , and performs the following ZKPKs:
 - (a) the opening of C_2 corresponds to the result of applying function f to the opening of commitment C_1 and string "1";
 - (b) \mathcal{U} possesses A 's signature on the opening of C_2 and C_3 (more precisely, only the parts of priv relevant for obtaining access to the resource);
 - (c) the opening of C_3 satisfies the access control rules imposed by \mathcal{S} .
5. \mathcal{S} verifies all proofs in step 4, and if they pass, grants user \mathcal{U} access to the resource.
6. The device erases all information captured and computed during the authentication process (in particular, it is important that the device erases W' and all information derived from it).

4.2 Security Analysis

In this section, we give a more detailed description of the security games and prove security of the scheme.

We model soundness as a game in which adversary \mathcal{A} is allowed to corrupt users, obtain access to their credentials cred , and engage in authentication protocols with the server, as well as monitor authentication protocols of other (honest) users in the system. Let $\mathcal{U}_1, \dots, \mathcal{U}_n$ denote the set of users that \mathcal{A} controls and let $\text{cred}_{\mathcal{U}_1}, \dots, \text{cred}_{\mathcal{U}_n}$ denote their corresponding credentials. \mathcal{A} wins the game if it is able to successfully authenticate to the server by forging credentials $\text{cred}'_{\mathcal{U}'}$ of a user \mathcal{U}' it does not control or by gaining access to more resources than what the corrupted users can already access (and possibly authenticating as one of the users it controls). An authentication scheme is sound if \mathcal{A} has at most negligible advantage in winning this game.

One way to model unlinkability is to have a simulator Sim that can simulate a valid execution of the authentication protocol without access to user information. Then no information about the user is leaked if the server's view after interacting with a valid user is indistinguishable from its view after interacting with the simulator. The unlinkability game proceeds as follows: the adversary \mathcal{A} represents the authority A colluding with the server \mathcal{S} . It creates the authority's private and public keys, enrolls all users, and possibly corrupts some users obtaining full access to their credential information (including random choices made at enrollment time). After that, \mathcal{A} engages in a challenge authentication protocol with either a valid uncorrupted user or a simulator. We say that unlinkability is achieved if, for any adversary \mathcal{A} , it is able to correctly guess whether it was communicating with a real user or a simulator with the probability at most negligibly larger than $1/2$.

To ensure biometric privacy, we let \mathcal{A} obtain access to user credentials. Then \mathcal{A} who is in possession of credentials $\text{cred}_{\mathcal{U}_1}, \dots, \text{cred}_{\mathcal{U}_n}$ should be unable to extract biometric information of any of the users with high probability and successfully authenticate (on behalf of one of them or as a non-existing user) without proper biometric data. This property is required to hold even if \mathcal{A} colludes with \mathcal{S} .

Before showing security of the overall scheme, we provide a supplemental result. Let $\sigma_A(m_1, \dots, m_\ell; r)$ denote CL-signature with A 's key on values m_1, \dots, m_ℓ using randomness r . That is, we make the random value used for hiding the messages explicit in the representation of the signature.

Lemma 1. *There exists a CL-signature scheme over group G such that, given a CL-signature $\sigma_A(m_1, \dots, m_\ell; r)$ and values $m_1, \dots, m_{j-1}, m_{j+1}, \dots, m_\ell$ for some $1 \leq j \leq \ell$ and r , it is infeasible for a polynomial-time adversary to determine m_j , where m_j is drawn from a distribution indistinguishable from uniform to a polynomial-time adversary, assuming that the discrete logarithm problem is hard. This holds even if the signing key sk_A is known.*

We omit the proof due to space considerations.

Theorem 2. *Assuming the security of the CL-signature scheme and the verifiable random function f , the scheme presented above is a secure anonymous biometric-based authentication scheme in groups where the discrete logarithm problem is hard.*

Proof. Completeness: this property can be shown to hold by examination.

Soundness: As previously described, we let adversary \mathcal{A} enroll in the system on behalf of users and corrupt existing users. For the set of users $\mathcal{U}_1, \dots, \mathcal{U}_n$ that \mathcal{A} controls, we

assume that not only \mathcal{A} has access to their credential information $cred_{\mathcal{U}_i}$, but can also obtain access to information not stored in the credentials. In particular, we assume that \mathcal{A} can have access to key K for each user it controls. Let $K^{(i)}$ denote the key belonging to user \mathcal{U}_i . To violate soundness of the scheme, \mathcal{A} then will either attempt to authenticate using invalid K' or using one of valid $K^{(i)}$ with privileges \mathcal{U}_i does not have.

In the first case, \mathcal{A} has to produce a proof that either it knows the authority's signature on openings of C_2 and C_3 or it has to correctly form the zero-knowledge proof in step 4(a) of the AC-Auth using K' of \mathcal{A} 's choice. The former can happen only with a negligible probability assuming that CL-signatures are secure, and the latter can also happen only with a negligible probability assuming that the function f is secure.

In the second case, \mathcal{A} attempts to authenticate using $K^{(i)}$ and privileges \mathcal{U}_i does not have. The only way for \mathcal{A} to achieve this is to produce a proof that it knows the authority's signature on openings of C_2 and C_3 , which can happen with at most negligible probability.

Unlinkability: Let \mathcal{A} be the adversary defined for the unlinkability game that represents the colluding \mathcal{A} and \mathcal{S} (and thus has access to enrollment information). During the challenge, \mathcal{A} is asked to engage in an authentication protocol with either a real user \mathcal{U} or a simulator Sim without access to any user information. Our simulator engages in the authentication protocol by performing the following steps:

1. Sim chooses K' at random, computes $K = f_{K'}("1")$, and produces commitments $C_1 = com(K'; z_1)$ and $C_2 = com(K; z_2)$.
2. Sim selects privileges $priv$ and computes commitment $C_3 = (priv, z_3)$.
3. Sim produces a zero-knowledge proof that the opening of C_2 corresponds to the result of applying f to the opening of C_1 , which we denote by π_1 .
4. Sim produces a simulated proof of knowledge, π_2 , of a CL-signature from the authority on the openings of C_2 and C_3 . This requires usage of the corresponding simulator of CL-signatures.
5. Sim produces a proof π_3 that the opening of C_3 satisfies the access control rules.

At the end of this interaction, \mathcal{A} obtains $(C_1, C_2, C_3, \pi_1, \pi_2, \pi_3)$. We next argue that \mathcal{A} 's view during interaction with a simulator is indistinguishable from its view when interacting with a real user.

The commitments C_1 , C_2 , and C_3 information-theoretically hide the values encoded in them, and therefore the values Sim chooses are indistinguishable from those chosen by real users. The proofs π_1 and π_3 produced by the simulator are real proofs of knowledge and thus are indistinguishable from a user's proofs. Finally, the (simulated) proof π_2 differs from a real proof of knowledge of a CL-signature, but due to the security of CL-signatures, \mathcal{A} can distinguish between a real and simulated proofs only with a negligible probability. Therefore, \mathcal{A} can distinguish between real and simulated protocol executions with at most negligible probability, as required.

Privacy of biometric: Let \mathcal{A} be in possession of credentials $cred_{\mathcal{U}_1}, \dots, cred_{\mathcal{U}_n}$. Since biometric information encoded in each user credential is independent of information included in credentials of other users, \mathcal{A} does not gain additional advantage in impersonating a user or recovering her biometrics by using information in other credentials, and we consider attacking a user by using only her own credentials. We have

$\text{cred}_{\mathcal{U}} = (\sigma_A(K, \text{priv}; z), z, P', V, \text{priv})$. For \mathcal{A} to impersonate the user, it has to either recover K from $\text{cred}_{\mathcal{U}}$ or authenticate without knowledge of K . Lemma 1 states that the former is infeasible when K is indistinguishable from a random value (which in our case is true due to pseudo-randomness of output of function f that we use to produce K), and the soundness property states that the latter is infeasible. To be able to recover \mathcal{U} 's biometric from the credentials, the only way for \mathcal{A} to do this is to recover the biometric (or other information that leads to recovery of the biometric) from P' . Assuming that a secure biometric generator is used to produce P' , \mathcal{A} can be successful only with a small probability. \square

5 Practical Considerations

The purpose of this section is to assess the feasibility of applying theoretical cryptographic techniques to empirical biometric data, as proposed in this work. We use iris codes as an example biometric. The two main questions we would like to address is (i) whether biometric key generation from iris codes is feasible, and (ii) whether key material extracted from iris codes can satisfy the requirements of the cryptographic tools.

In regards to the first question, the work of Hao et al. [23] was able to achieve a notable step toward biometric key generation by constructing a secure sketch for iris data. The construction used nested error-correcting codes and was able to achieve excellent key recovery rates. In particular, it used Hamming distance over binary strings as the metric with two types of error-correcting codes. A disadvantage of any secure sketch based approach is that correcting a large number of errors can cause the public data to potentially leak a lot of information about the biometric. In particular, as described in section 3, the current techniques give only worst-case information leakage analysis, which is measured as the loss of entropy after the release of the public data. Then to correct 10% of errors in a 2048-bit iris code, up to 411 bits of entropy can potentially be leaked. An iris code, however, is estimated to have about 250 degrees-of-freedom [18], and a noticeably higher error rate for authentic codes must be tolerated.

A natural way to lower the entropy loss in this case is to attempt to reduce the noise. A pioneering work on biometric-based authentication of Davida et al. [19] gave the idea of reducing the noise by acquiring multiple samples and performing majority decoding to create a single image with low noise. That is, during both the enrollment and key recovery stages, the algorithms take a number of biometric readings W_1, W_2, \dots, W_m rather than a single one, and create a single representation W using majority computation that more accurately represents the corresponding biometric. This technique is believed to be effective, and was recently empirically evaluated on iris codes in [4]. For a realistic bound of error tolerance of 30% [35,23], the theoretical approach significantly reduces the error (e.g., to 5% if 15 scans were used), while in practice the error was shown not to go lower than 16–18% [4], which is still a significant improvement. Other techniques for reducing noise also exist (e.g., scanning both eyes instead of a single one for iris recognition), which can be combined to result in even lower error rates. This means that the entropy loss will become tolerable when the error rate is reduced to a rather small value. We emphasize that the entropy-based analysis is only an upper

bound on the information leakage that we can compute rather than a close estimate, and the latter is likely to be significantly lower.

Now with respect to the second question raised above, we have already seen that iris codes can be assumed to have at least 250 bits of entropy. Currently known constructions of extractors extract all of the randomness from a source [31] assuming a sufficient number of additional truly random bits (which in our case are supplied as a part of the public information in sufficient quantity). This means that we can extract a 250-bit random string from an iris code. Our construction is built using groups over elliptic curves with bilinear pairings for which using a 160–190-bit modulus is sufficient. Thus, a random string extracted from a biometric has sufficient amount of randomness, and we will be able to construct the key exactly as prescribed by the protocol.

In addition to the above concerns, recent work of Simoons et al. [33] shows that existing constructions of secure sketches and the keys produced by fuzzy extractors that rely on them might not meet security requirements sought of encryption keys. In particular, the release of public data might allow one to link together ciphertexts belonging to the same individual (i.e., generated using keys produced from related biometrics). This threat, however, does not exist in our framework because the public data always stays with the client and the authentication protocol does not leak any information that can be linked to the individual.

6 Related Work

Related work on biometric-based key generation is very extensive, especially publications at biometric-related venues, and its survey is beyond the scope of this work. Anonymous credentials where biometric data are used for non-transferability are known in prior literature. They were first introduced by Bleumer in [5], and later expanded upon by Impagliazzo and More in [24]. The construction of Bleumer is based on the “Wallet with Observer” protocol originally constructed by Chaum and Pedersen for achieving the properties of non-transferability and privacy-protection. The second paper extends and formalizes these results, and adds the feature of revoking credentials. Such “Wallet with Observer” architecture, which is used in both of these works, however, requires non-trivial tamper-resistant hardware that runs trusted processes and executes parts of cryptographic protocols.

Other approaches to achieving non-transferability include encoding external sensitive information, e.g., credit card numbers, into the credentials (see, e.g., [11]). Then, when such credentials are used, the owner must prove knowledge of said information to the verifier. While such information may be shared by close relatives or friends who will be able to use the credentials on behalf of the owner, such sharing is often acceptable to service providers, and their goal is to prevent large-scale sharing of credentials (by, e.g., posting credentials on a Web page). The above approach assumes that the credential issuer will have access to a large quantity of external or otherwise sensitive information about the user that can be included in the credential. When, however, such information is not readily available, alternative solutions must be sought. The current framework provides such an alternative.

7 Conclusions

This work examines most researched techniques for extracting keys from biometric data and their use in cryptographic applications. We first provide a generic mechanism of enhancing biometric privacy protection of biometric key generators. We then build on biometric key generators to construct anonymous credentials, where biometric is used to ensure non-transferability of user credentials. Unlike previous proposals, we target at making minimum trust assumptions on the execution environment: we only require a fresh biometric to be captured on each use of such credentials. Then even if tampering with the user device results in full access to the information stored on it, this does not lead to weakening the security guarantees nor compromises the biometric, even in the event of collusion of multiple participants.

The scope of this work could not cover all schemes for biometric key generation and could not provide their thorough analysis with respect to security and re-usability. Thus, we leave it to future work to analyze the security of different BKG constructions and different metric spaces and determining which constructions would provide the best security guarantees for a particular type of biometric data.

Acknowledgments

This work benefited from discussion regarding biometrics with Karen Hollingsworth, Patrick Flynn, and Kevin Bowyer.

References

1. Arakala, A., Jeffers, J., Horadam, K.: Fuzzy extractors for minutiae-based fingerprint authentication. In: Lee, S.-W., Li, S.Z. (eds.) *ICB 2007*. LNCS, vol. 4642, pp. 760–769. Springer, Heidelberg (2007)
2. Bakhtiari, A., Shirazi, A., Zamanlooy, B.: An efficient biocryptosystem based on the iris biometrics. In: Mery, D., Rueda, L. (eds.) *PSIVT 2007*. LNCS, vol. 4872, pp. 334–345. Springer, Heidelberg (2007)
3. Ballard, L., Kamara, S., Reiter, M.: The practical subtleties of biometric key generation. In: *USENIX Security Symposium*, pp. 61–74 (2008)
4. Blanton, M., Aliasgari, M.: Secure computation of biometric matching. Technical Report 2009–03, Department of Computer Science & Engineering, University of Notre Dame (2009)
5. Bleumer, G.: Biometric yet privacy protecting person authentication. In: Aucsmith, D. (ed.) *IH 1998*. LNCS, vol. 1525, pp. 99–110. Springer, Heidelberg (1998)
6. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
7. Boyen, X.: Reusable cryptographic fuzzy extractors. In: *ACM Conference on Computer and Communications Security (CCS 2004)*, pp. 82–91 (2004)
8. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)
9. Bresson, E., Stern, J.: Proofs of knowledge for non-monotone discrete-log formulae and applications. In: Chan, A.H., Gligor, V.D. (eds.) *ISC 2002*. LNCS, vol. 2433, pp. 272–288. Springer, Heidelberg (2002)

10. Bringer, J., Chabanne, H., Cohen, G., Kindarji, B., Zemor, G.: Optimal iris fuzzy sketches. In: IEEE BTAS, pp. 1–6 (2007)
11. Camenisch, J.L., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
12. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
13. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
14. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
15. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report No. 260, ETH Zurich (1997)
16. Chaum, D., Evertse, J.-H., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)
17. Clancy, T., Kiyavash, N., Lin, D.: Secure smartcard-based fingerprint authentication. In: ACM SIGMM Workshop on Biometrics Methods and Applications, pp. 45–52 (2003)
18. Daugman, J.: How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology* 14(1), 21–30 (2004)
19. Davida, G., Frankel, Y., Matt, B.: On enabling secure applications through off-line biometric identification. In: IEEE Symposium on Security and Privacy, pp. 148–157 (1998)
20. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal of Computing* 38(1), 97–139 (2008)
21. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
22. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
23. Hao, F., Anderson, R., Daugman, J.: Combining crypto with biometrics effectively. *IEEE Transactions on Computers* 55(9), 1081–1088 (2006)
24. Impagliazzo, R., Miner More, S.: Anonymous credentials with biometrically-enforced non-transferability. In: ACM Workshop in Privacy in the Electronic Society (WPES 2003), pp. 60–71 (2003)
25. Juels, A., Sudan, M.: A fuzzy vault scheme. In: International Symposium on Information Theory (2002)
26. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: ACM Conference on Computer and Communications Security, pp. 28–36 (1999)
27. Lee, S., Moon, D., Jung, S., Chung, Y.: Protecting secret keys with fuzzy fingerprint vault based on a 3d geometric hash table. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007. LNCS, vol. 4432, pp. 432–439. Springer, Heidelberg (2007)
28. Lee, Y.J., Bae, K., Lee, S.J., Park, K.R., Kim, J.: Biometric key binding: Fuzzy vault based on iris images. In: Lee, S.-W., Li, S.Z. (eds.) ICB 2007. LNCS, vol. 4642, pp. 800–808. Springer, Heidelberg (2007)

29. Nagar, A., Chaudhury, S.: Biometrics based asymmetric cryptosystem design using modified fuzzy vault scheme. In: International Conference on Pattern Recognition (ICPR 2006), pp. 537–540 (2006)
30. Nandakumar, K., Jain, A., Pankanti, S.: Fingerprint-based fuzzy vault: Implementation and performance. *IEEE Transactions on Information Forensics and Security* 2(4), 744–757 (2007)
31. Nisan, N., Ta-Shma, A.: Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences* 58, 148–173 (1999)
32. Pedersen, T.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
33. Simoens, K., Tuyls, P., Preneel, B.: Privacy weaknesses of biometric sketches. In: IEEE Symposium on Security and Privacy, pp. 188–203 (2009)
34. Uludag, U., Pankanti, S., Jain, A.K.: Fuzzy vault for fingerprints. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 310–319. Springer, Heidelberg (2005)
35. Uludag, U., Pankanti, S., Prabhakar, S., Jain, A.: Biometric cryptosystems: Issues and challenges. *Proceedings of the IEEE* 92(6), 948–960 (2004)
36. Yang, S.: Secure fuzzy vault based fingerprint verification system. In: Asilomar Conference on Signals, Systems, and Computers, vol. 1, pp. 577–581 (2004)
37. Yang, S., Verbauwhede, I.: Automatic secure fingerprint verification system based on fuzzy vault scheme. In: ICASSP, pp. 609–612 (2005)

Secure Remote Execution of Sequential Computations

Ghassan O. Karame, Mario Strasser, and Srdjan Čapkun

ETH Zurich, Switzerland
{karameg,capkuns}@inf.ethz.ch,
strasser@tik.ee.ethz.ch

Abstract. We describe a scheme that secures the remote execution of sequential computations in grid-computing scenarios. To the best of our knowledge, this is the first contribution that addresses the security of generic sequential computations. By dividing sequential tasks into smaller subtasks and permuting them among participants, we show that our scheme facilitates the insertion of selective redundancy and/or pre-computed functions (*ringers*) that are indistinguishable from other computations. We analyze the security of this proposal and we demonstrate that our scheme enables the detection of individual and colluding malicious participants. In addition, we show that our scheme can be equally used to securely track the progress of remote execution. We further investigate the damages introduced by possible chaining of errors within the remote execution and we discuss recovery mechanisms to counter these challenges. We validate our findings both analytically and empirically via simulations.

1 Introduction

The recent years have witnessed an increasing development of distributed computing platforms that leverage on the idle computing power of volunteer hosts (SETI@home [1], Distributed.net [2]) to run computationally expensive tasks on behalf of academic research groups, industry labs and even individual clients.

However, the open and untrusted environment in which these distributed computations are performed tends to cast suspicion on the results returned by the participants of these platforms. For instance, it is often the case that a participant (the task supervisor), that wishes to run a computationally expensive task, recruits several other nodes that agree to execute the task on its behalf in exchange of some form of reward (e.g., monetary remuneration). In typical scenarios, the supervisor has a limited computational capability and, therefore, cannot afford to check the computations itself.

One important challenge to address therefore is designing secure and efficient mechanisms to check the correctness of remote computations with minimum verification overhead. Although there is a large body of studies that address the security of distributed systems, there are few contributions that deal with the security of remote computations (e.g., [4], [5], [7]). With the exception of reputation/voting-based solutions [15], [16], [17], [18], an even smaller subset of these works proposes practical and efficient solutions to securely verify the execution of *generic* classes of sequential computations [4]. The latter class mainly refers to those sequential functions used in practice that are not

necessarily repetitive, yet too computationally expensive to be run on the supervisor’s machine (e.g., extensive simulations). The main challenge in securing these functions resides in the direct relation between intermediate results which might limit the number of viable countermeasures that can be used to prevent possible misbehavior during execution. Namely, the sequential nature of these computations prevents the use of selective embedded “security checks” (e.g., pre-computed functions or ringers [4]) within the flow of computations. Currently, the only known method to efficiently verify (with low overhead) remote sequential computations is by relying on sample redundancy (e.g., [5]). However, this method is not entirely secure against collusion between malicious entities.

In this work, we propose a probabilistic scheme that secures *sequential* computations, in spite of collusion among malicious hosts. To the best of our knowledge, this is the first contribution that addresses this problem and that demonstrates that the ringer scheme – initially proposed by Golle *et al.* in [4] to secure parallel computations – can be equally used to efficiently secure their sequential counterpart. Another important aim of our work is to provide a practical framework that enables the supervisor to efficiently recover from possible misbehavior in the execution of its tasks. The major challenge in executing sequential computations is that a single erroneous intermediate computation renders the results of the entire sequential task useless to the supervisor. While several contributions rely on the use of selective sampling to check the credibility of remote hosts, as far as we are aware, no prior work has tackled the problem of efficiently recovering from undetected erroneous computations. In this paper, we address this issue and we show that by capitalizing on the high detection rate of our scheme, the correctness of the sequential functions can be ensured with a modest overhead in execution time. Our contributions in this work are summarized as follows:

- We show that by breaking each task into smaller pieces and by permuting the resulting subtasks among different participants, the supervisor can efficiently make use of indistinguishable pre-computed functions (or “ringers” [4]), combined with selective redundancy to probabilistically secure and track the remote executions of its sequential tasks. We validate our findings both analytically and empirically via extensive simulations.
- We evaluate the robustness of our proposal in practical settings and we discuss efficient solutions that enable the supervisor to recover from possible tampering with the execution of its tasks. We further analyze the resilience of our scheme to chaining of errors caused by incorrect intermediate results.

The remainder of this paper is organized as follows. In Section 2 we briefly overview the related work in the area. In Section 3 we present our solution and we analyze its resilience against individual and colluding malicious participants. In Section 4 we analyze efficient solutions that enable the supervisor to remotely ensure the correctness of its executing tasks and to counter possible chaining of errors caused by intermediate incorrect computations. In Section 5 we discuss further insights with respect to our scheme and we conclude the paper in Section 6.

2 Related Work

A comprehensive survey in the area of result-checking and self-correcting programs can be found in [6]. Golle *et al.* propose in [4] to secure a specific class of parallel problems: inverse one-way functions (IOWF), where participants are required to compute the pre-images of several one-way functions. Their solution relies on inserting special-purpose *ringers* in each task. These correspond to the pre-computed images of randomly chosen elements. In [5], Szajda *et al.* extend this solution to secure non-sequential optimization problems and Monte Carlo simulations. In addition, the authors briefly propose assigning selective redundancy to secure repetitive sequential functions. Golle *et al.* further propose in [7] a security framework for commercial distributed computations that equally relies on selective redundancy. They further analyze the impact of varying the distribution that dictates the application of redundancy among participants. Similarly, Du *et al.* discuss in [10] a scheme that uses sampling techniques along with a Merkle-tree based commitment technique to secure non-sequential distributed problems in grid computing. Goodrich *et al.* discuss in [11] mechanisms to duplicate tasks among participants in grid computing applications as a mean to efficiently counter collusion among malicious participants.

Sanders *et al.* suggest computing with encrypted functions to provide security for mobile agents [8]. The major drawback of this proposal lies, however, in the fact that it might be very difficult to create encryption functions that result in correct executable procedures. Vigna *et al.* propose a mechanism based on execution tracing to protect the execution of mobile agents [9]. In [12], Yang *et al.* describe a method that uses the program counter values to monitor remotely executing computations. The supervisor, then, checks sample computations in order to detect possible misbehavior.

Several other proposals suggest the use of tamper-proof hardware/software [13] to prevent possible tampering with the results of the computations. However, tamper-proof software/hardware comes at high implementation costs nowadays. Another solution to the problem we consider in this paper would be for the supervisor to send an obfuscated executable code; however, existing code obfuscation techniques can only result in modest, best-effort efficacy nowadays [14].

3 Secure Verification and Tracking of Remote Execution

In this section, we describe our scheme that enables secure verification and tracking of the execution of sequential computations. and we analyze its resilience against individual and colluding malicious participants.

3.1 System and Attacker Model

Our computing platform consists of a *supervisor* interested in remotely running *several* sequential *tasks* (e.g., exhaustive simulations) on the machines of multiple *participants*. We assume that participants have considerable incentives to execute the tasks on behalf of the supervisor (e.g., in exchange of recognition, monetary reward). Detailed analysis of such incentive mechanisms is beyond the scope of this paper.

Individual tasks are independent of each others. However, each task can be divided into smaller sub-tasks that can be solved in a reasonable amount of CPU time. That is, owing to its sequential nature, a task function $f(x) = (g \circ h \circ j \circ k \dots)(x)$ can be divided into the sequential individual components $g()$, $h()$, etc.. The supervisor can directly extract these sub-functions from the original code (e.g., subroutines) or, alternatively, the supervisor can use available tools that decompose a code piece into smaller sub-routines according to its control flow structure. Note that our analysis equally covers the case of multiple inputs per function. We further assume secure communication between the supervisor and the participants and we abstract away the peculiarities of the communication channel, such as delays, congestion, jitter, etc..

We assume the existence of one or multiple malicious participants. These participants possess technical skills by which they can efficiently analyze, decompile, and modify executable code as necessary. Furthermore, these participants have knowledge of the measures used by the supervisor to prevent potential tampering with the computations.

In our analysis, we assume that malicious participants are motivated to cheat in order to obtain credit without performing all of their assigned work. For instance, a selfish participant might only execute 50% of its assigned job and defect from running the rest of its tasks. In the mean time, it might decide to use its resources to run *another* task by a different participant to increase its benefit in the network. Here, two or more malicious participants might collude to increase their chances of not being detected.

Similar to [4], [5], [10], we do not consider the case where a malicious participant cheats only once in an attempt to disturb the computations (for personal gain or to achieve some e.g., political goal). To the best of our knowledge, with the exception of re-checking every subtask or relying on tamper-resistant software, little can be done by the supervisor to counter this threat. In this work, we assume, however, that untrusted participants are motivated to cheat in a considerable number of subtasks (e.g., > 10%).

3.2 Securing the Remote Execution of Sequential Tasks

We assume that the supervisor is interested in executing N distinct and independent tasks on the remote machines of P different participants. Our scheme described hereafter can be applied to arbitrarily chosen N and P such that $N \leq P$. Our scheme for securing the remote execution of N tasks unfolds as follows:

1. The supervisor first divides each task into M smaller subtasks. This can be achieved by decomposing the composite function of the task into its smaller functional components. Alternatively, the subtasks could be obtained by extracting the control flow structure of the task function. Note, here, that the subtasks do not necessarily have to be of the same computational length. The supervisor then proceeds to running the N tasks on the machines of P participants in M consecutive rounds as shown in Figure 1.
2. In round i , the supervisor picks an idle participant and according to some probability, it decides to verify its credibility by inserting “security checks” within the computations; otherwise it randomly assigns to the participant a pending subtask. In our scheme, the supervisor evaluates the credibility of a participant by requesting that it runs a subtask whose results are already known to the supervisor (a *ringer*) or by redundantly assigning the same subtask to another participant. We show later that

this process is transparent to participants and that they cannot distinguish whether they are running a legitimate subtask or whether their work is being checked by the supervisor. Round i ends when all N participants are assigned to a job. In this way, the supervisor checks the work of several participants in each round.

3. At the beginning of round $i + 1$, the supervisor collects the results reported by the participants and checks the correctness of the ringers and the redundantly assigned subtasks. If these verifications pass, the supervisor re-permutes the next logical subtasks (since each task is sequential) among the participants that are considered to be honest while using the corresponding outputs of the last round as inputs to the subtasks of this round. Otherwise, if the supervisor detects inconsistent results, it stops interacting with the malicious node¹.
4. The supervisor repeats Steps 2) and 3) until all subtasks are executed.

Before analyzing the security of our proposal, we first describe the main rationale behind our approach.

The Main Intuition: The main intuition behind our proposal relies on the fact that by randomly permuting the execution of N different tasks, the supervisor gains a considerable advantage in securing their remote execution when compared to the scenario that features a single executing task.

The major challenge in securing sequential tasks resides in the fact that the output of one subtask is the input to the next consecutive subtask. This limits the number of viable “tricks” that the supervisor can make use to prevent participants from cheating. Note that redundant computations might not solely achieve a desired level of security in scenarios where collusion between malicious participants is possible. In this case, the supervisor might have to accept the burden of checking sample computations itself.

By permuting N different tasks among P participants (Figure 1), the

sequential property of the tasks becomes largely transparent to the participants. This enables the supervisor to embed *pre-computed* indistinguishable “checks”, *ringers*, within the subtasks assigned to participants. Ringers were first proposed in [4] to secure a

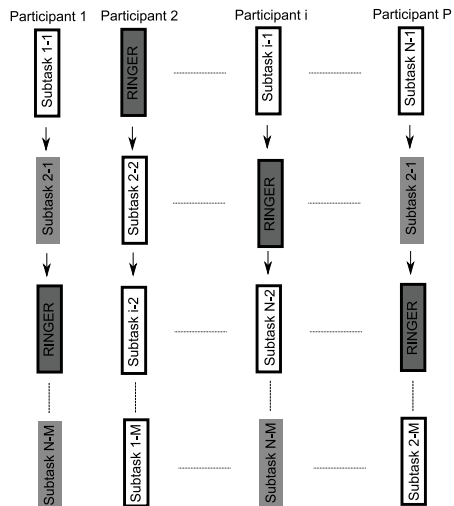


Fig. 1. N tasks assigned to P participants. Each box denotes a subtask. The notation “Subtask X-Y” refers to subtask # Y of task # X. The dark gray boxes refer to “ringer” subtasks (pre-computed subtasks whose result is known to the supervisor) and the boxes with shaded borders refer to redundantly assigned subtasks.

¹ It is out of the scope of this paper to discuss mechanisms to isolate malicious participants from the network. The supervisor can make use of cryptographic proofs to inform a central authority of its opinion about participants (e.g., [20]).

special class of non-sequential inverse one-way functions. Provided that ringers are indistinguishable to participants from other subtasks, they provide a strong form of probabilistic security to remote computations. In fact, since ringers can be directly verified by the supervisor, they are resilient to collusion between malicious participants and do not incur computational burden on the supervisor, which makes them ideal for real-time grid-computing applications.

Generating Indistinguishable Ringers: In our scheme, ringers could be constructed from previously executed tasks. We point that a poor choice of ringer candidates might allow malicious colluding participants to abuse our scheme. If ringer subtasks were completely independent from each other, then malicious participants can collude and analyze the input/output of each subtask they execute: if the input of one subtask was provided previously by another participant, colluding participants could identify that the subtask in question is not a ringer subtask and that it pertains to a genuine task. As mentioned previously, if the supervisor uses previously executed tasks (or any “reasonable” executable code) as ringer tasks, the corresponding ringer subtasks are likely to share comparable computational workload when compared to other subtasks while inherently exhibiting a similar relation between its inputs and outputs. This would indeed ensure that the inserted ringers cannot be distinguished from other subtasks in the system, in spite of collusion among malicious participants.

Let the random variable X denote the number of subtasks pertaining to the same task run by the same participant. In our scheme, the probability that a participant runs *at most* one subtask pertaining to each task ($X \leq 1$) in M execution rounds is:

$$P[X \leq 1] = \sum_{i=0}^{i=1} \binom{M}{i} \left(\frac{1}{N}\right)^i \left(1 - \frac{1}{N}\right)^{M-i}, \quad (1)$$

where N is the number of required tasks to be run. In Figure 2 we plot $P[X \leq 1]$ with respect to different values of N . Equation 1 suggests that by randomly permuting tasks and their corresponding subtasks among the participants in the system, the probability that a participant cannot distinguish which task it is actually running in each execution round is satisfactorily large (> 0.7) given a reasonable number of participants and tasks in the system ($N \leq P$). This suggests that it is highly unlikely for participants to establish a correlation between different tasks. Since participants do not know the number of tasks in the system, the supervisor can take advantage of this fact and probabilistically requests that participants run *ringer subtasks*. Recall that once the supervisor detects a malicious participant, it stops interacting with the detected node.

To ensure an acceptable level of security, the number of embedded ringers should be considerable when compared to the number of subtasks executed per participant (typically $> 20\%$). However, depending on the nature of the supervisor tasks (e.g., repetitive tasks such as in the GIMPS project [3]), finding a large number of ringers might be prohibitively expensive [5]. *To address this issue, the supervisor can combine the use of ringers along with sample redundancy.* For example, if the supervisor needs to randomly check 40% of a participant’s work and possesses a number of ringer candidates that only account for 20% of the number of subtasks per participant, then the supervisor can redundantly assign 20% of the subtasks to achieve its desired level of security.

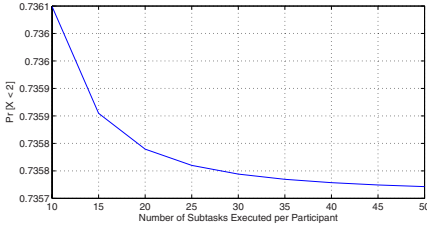


Fig. 2. Probability that a participant runs at most one subtask pertaining to each task. Here, we assume that the number of subtasks per task is equal to the number of available tasks.

M	P_C	P_R	P_P	P_M	Overhead	P
10	0.7	0.15	0.15	0.2	30%	0.88
10	0.9	0.15	0.15	0.2	30%	0.94
20	0.5	0.15	0.15	0.2	30%	0.95
20	0.5	0.25	0.25	0.2	50%	0.99
20	0.5	0.15	0.15	0.4	30%	0.92
20	0.5	0.2	0.1	0.4	30%	0.94
50	0.5	0.2	0.2	0.2	40%	0.999

Fig. 3. Probability of detecting malicious participants with respect to different input parameters. M is the the number of sub-tasks per task.

We show in the following subsection that this combination remains, to a large extent, resilient to collusion between malicious participants.

Coordination and Synchronization Overhead: Similar to [5], our scheme incurs an increased workload on the supervisor since it has to coordinate the permutation of sub-tasks among participants “on the fly”. Given a modest number of tasks and participants (typically < 50), this process can be made very efficient in real-time settings through an automated interface that coordinates the computations between participants. In Section 3.4, we discuss in greater details the benefits of this solution. Furthermore, subtask permutation might require loose synchronization among the participants’ machines to efficiently manage the time cost of computations. However, since the supervisor will presumably pick those participants with considerable computing performance, synchronization costs are likely to be satisfactorily negligible. Note that the synchronization costs in our scheme can be further minimized if the supervisor makes use of an optimal assignment of subtasks to participants that optimizes the total execution time. In Section 4.1, we show via simulations that our proposed scheme performs well even in the case when the subtasks are assigned to participants at random. Note that our scheme does not induce any additional communication overhead on the supervisor; the supervisor would have the same communication burden even if it would send the full tasks to the participants.

3.3 Security Analysis

Throughout our analysis, we make the worst case assumption that malicious participants will *always* collude with each other. That is, if the supervisor redundantly assigns subtasks to two or more malicious nodes, we assume that they will coordinate their results to avoid being detected. We further assume that the distribution of ringers and redundant subtasks is uniform per subtask. This suggests that the best strategy of a malicious participant to decrease the likelihood that it gets caught is to equally follow a uniform distribution of cheating per subtask. In case the supervisor detects inconsistencies between the results of the redundantly assigned subtasks without being able to determine the genuine subtasks’ outcome (i.e., with the absence of majority consensus), we assume in our analysis that it will proceed to re-run the corresponding subtasks

(Section 4) to increase its confidence in their correctness. We do acknowledge that there might exist methods to redundantly assign subtasks among participants that might perform better than uniform distributions [11]; however, we show in this work that even the simplistic random redundancy achieves a satisfying level of security when combined with ringers. We point that the supervisor is not likely to benefit by choosing biased distributions in verifying the job of participants since these latter might change their cheating strategies accordingly in order to increase their gain. We, nevertheless, briefly discuss in Section 5 the implications of choosing non-uniform distributions on the performance of our scheme.

On the other hand, in our scheme, assuming that a malicious participant cheats with probability P_C per subtask and that the supervisor inserts an indistinguishable ringer with probability P_R per subtask or redundantly assigns the same subtask to another node with probability P_P , then the probability P_T of catching a malicious participant in a single subtask is given by: $P_T = P_C(P_R + P_P(1 - P_M))$. Here, subtask redundancy will only be beneficial when the supervisor picks an honest participant (with probability $(1 - P_M)$, where P_M is the fraction of malicious nodes in the network).

Assuming that each participant is required to execute at least M different subtasks, then the probability that the supervisor catches potential misbehavior in our scheme is computed as follows:

$$P = 1 - (1 - P_T)^M = 1 - (1 - P_C(P_R + P_P(1 - P_M)))^M. \tag{2}$$

Table 3 shows the probability of detecting a malicious participant with respect to various input parameters. As P_C increases, the probability to detect a malicious participant equally increases; in the case where malicious participants always cheat in their subtasks, the probability that they get detected in our scheme approaches 1. Note that the higher is the number of inserted ringers and/or redundant computations, the higher is the level of assurance in the correctness of the computations and the higher is the induced time overhead of our scheme. In other words, the time required for the computations to complete increases by the amount of time needed to execute all the inserted ringers

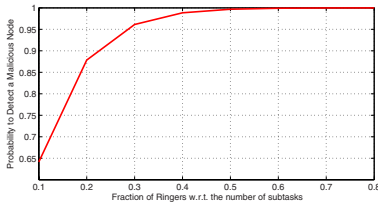


Fig. 4. Probability to detect cheating versus the fraction of inserted ringers. Here, the number of subtasks is 20. The network contains 20% malicious participants that cheat in 50% of their assigned subtasks.

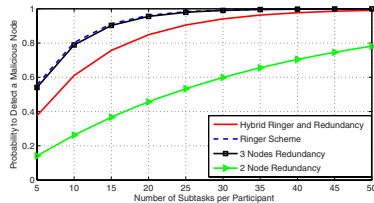


Fig. 5. Comparison of probabilistic schemes for securing sequential computations. Here, we assume that the supervisor randomly checks 30% of the subtasks. We consider scenarios where 20% of the participants are malicious and randomly cheat in 50% of the subtasks.

and redundant subtasks. Nevertheless, as shown in Figure 4 our scheme can ensure an acceptable security level even when the overhead resulting from inserting ringers and redundant computations is as low as 30%.

In Figure 5 we analytically compare the security of ringer-based schemes with solutions based on selective redundancy with respect to the number of subtasks per participant. Our findings suggest that ringer-based solutions considerably outperform redundancy-based schemes in scenarios where malicious participants collude. For instance, when the supervisor selectively checks 30% of the computations performed by a participant, by solely relying on ringers, the probability of detecting a malicious participant is as high as selectively assigning redundant subtasks to 3 different participants. Furthermore, even in hybrid solutions – equally combining ringer-schemes and selective redundancy – the probability of detecting possible cheating is at least twice as high when compared to redundancy-based solutions.

3.4 Secure Tracking of Remote Execution

As described earlier, the starting time of the computations required for each subtask is known to the supervisor. Since subtask permutation and verification is performed on the fly, each participant depends on the supervisor to acquire its newly assigned subtask function along with its corresponding input. This equally allows the supervisor to know the time each subtask took to complete; as shown in Figure 6, once a participant completes the execution of a subtask, it reports the result back to the supervisor, which then verifies the reported result (in case the subtask corresponded to a ringer or it was redundantly assigned) and sends back to the participant another subtask to execute.

Given the random permutation, participants cannot guess beforehand which subtask they are going to execute. This suggests that the starting time of each individual computation is solely dependent on the supervisor. Malicious participants could, still, try to trick the supervisor by reporting incorrect results. As explained previously, such misbehavior will be detected by the supervisor with high probability. Malicious participants could equally delay reporting the results to the supervisor. However, given that the participants are rational players aiming at maximizing their benefit in the network (e.g., claim credit for their work), participants are unlikely to benefit from this strategy.

We conclude that our scheme enables probabilistically secure tracking of remote execution of the supervisor tasks at a subtask granularity. That is, the supervisor knows at anytime during the execution process the number of executed (and pending) subtasks in each task.

4 Ensuring the Correctness of the Sequential Tasks

In the previous section, we analyzed the performance of our scheme in detecting malicious participants. In what follows, we evaluate other practical aspects and limitations related to our proposal.

From a practical perspective, one potential limitation of our scheme lies in the fact that a single undetected intermediate result renders the entire’s task output erroneous; this limitation applies to all solutions that target sequential computations. Furthermore,

due to task-permutation, one malicious participant is likely to cheat in several subtasks pertaining to different tasks before getting detected. This might result in the deterioration of the system's efficiency. In what follows, we analyze these limitations and we discuss efficient solutions to counter their impact.

Assuming a perfect random permutation of tasks among participants such that each participant runs at most a single subtask of each task and given that a malicious participant cheats with probability P_C per subtask, then the maximum fraction of incorrect tasks in the system T is computed as follows:

$$T = P_C(1 - (P_R + P_P(1 - P_M))), \quad (3)$$

where P_R and P_P denote the probability of inserting a ringer and redundant assignment per subtask respectively and P_M is the initial fraction of malicious nodes in the network.

The most intuitive solution to limit the damage that a malicious participant can cause is by increasing the number of checks (e.g., ringers, redundant assignment). This also comes at the benefit of increasing the confidence C in the correctness of the subtasks' results that were previously computed by a participant. C is computed as follows:

$$C = 1 - T = 1 - (P_C(1 - (P_R + P_P(1 - P_M)))). \quad (4)$$

The drawback of an increased confidence C is a prolonged task execution time; once the supervisor detects that a participant is malicious, it has to re-run all the tasks that the malicious participant participated in executing. This is needed to prevent a possible chaining of errors due to the sequential property of the tasks.

In what follows, we analyze this additional cost as a function of the confidence in the correctness of the tasks. More precisely, we compute the number of subtask executions (i.e., rounds) that are required to correctly complete a task, given an initial fraction of malicious nodes and the verification overhead per task (i.e., P_R and P_P).

Recall that detected malicious participants are isolated and no longer considered for subtask execution (in Section 5 we discuss the case where the supervisor might not be able to fully isolate malicious nodes). The fraction of malicious participants therefore decreases over time upon every detection. More specifically, let P_{M_i} be the fraction of malicious participants in the i -th round. Given that all N tasks can be executed in parallel (i.e., that $P \geq N$), the expected number of malicious participants that are newly detected in the i -th round is $NP_{M_i}P_C(P_R + P_P(1 - P_{M_i}))$. That is, after $i - 1$ rounds, a total of Q_{i-1} malicious nodes have been removed:

$$Q_{i-1} = \sum_{j=0}^{i-1} NP_{M_j}P_C(P_R + P_P(1 - P_{M_j})). \quad (5)$$

Hence, the expected fraction of malicious nodes in the i -th round is:

$$P_{M_i} = \frac{P_M P - Q_{i-1}}{P - Q_{i-1}} = \frac{P_M P - \sum_{j=0}^{i-1} NP_{M_j}P_C(P_R + P_P(1 - P_{M_j}))}{P - \sum_{j=0}^{i-1} NP_{M_j}P_C(P_R + P_P(1 - P_{M_j}))}. \quad (6)$$

The expected number of rounds i^* after which all malicious participants have been identified and isolated can now be derived by (numerically) solving the equation $\lfloor P_M P - \sum_{j=0}^{i-1} N P_{M_j} P_C (P_R + P_P (1 - P_{M_j})) \rfloor = 0$ for i . The *upper bound* $\overline{i^*}$ on the expected number of rounds to complete a task is computed as follows:

$$\overline{i^*} = i^* + M.$$

This suggests that after i^* rounds all the malicious nodes are eliminated; the final result can then be computed in M rounds. Figure 7 depicts this upper bound as a function of the initial fraction of malicious nodes (P_M) and the verification overhead (P_R and P_P).

To obtain a more accurate estimate of the expected running time of a task as a function of the confidence in its correctness, we model its execution with an absorbing Markov chain [19] (refer to Figure 6). In this chain, the states $s_{i,j}$ represent the number j of subtasks that have already been executed after i rounds and each state transition accounts for one subtask execution. We further associate to each state $s_{i,j}$ a random variable $Y_{i,j}$ that represents the number of required state transitions (i.e., rounds) to reach an absorbing state from $s_{i,j}$; once an absorbing state is reached, the processing of a task terminates.

For example, consider the situation represented by state $s_{2,1}$. Here, after two rounds only the result of the first subtask has been accepted. As depicted in Figure 6, there essentially exists two possibilities on how the processing of the task can proceed:

1. If neither the currently used participant nor the participant that computed the previous subtask are identified as malicious in this round, both results are considered valid. This new state (two subtasks completed after three rounds) is represented by state $s_{3,2}$. From the analysis in Section 3.3, it follows that in round i a participant is identified as malicious with probability P_{Z_i} :

$$P_{Z_i} = P_{M_i} P_C (P_R + P_P (1 - P_{M_i})). \quad (7)$$

The probability that we proceed to state $s_{3,2}$ (i.e., that none of the $j + 1 = 2$ participants we used so far has been identified as malicious) is thus given by $(1 - P_{Z_i})^{(j+1)} = (1 - P_{Z_i})^2$.

2. If at least one of the nodes that participated in the execution of the completed subtasks is identified as malicious, all subtasks subsequent to those executed by a malicious participant must be discarded. In state $s_{2,1}$, the likelihood of this event is

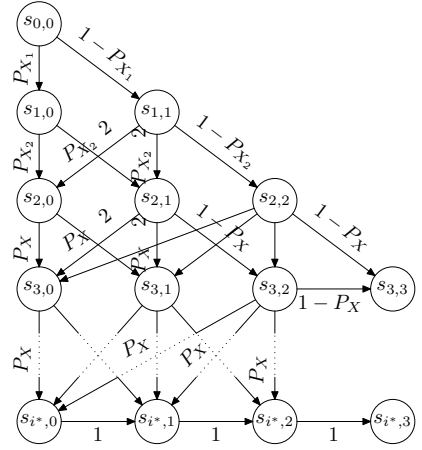


Fig. 6. Markov chain representing a task execution for $M = 3$. The states $s_{i,j}$ represent the number j of subtasks that have been executed after i rounds. We omit some labels for purposes of better clarity.

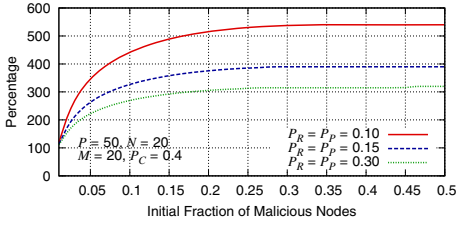


Fig. 7. Upper bound on the expected number of rounds to complete a task as a function of the initial fraction of malicious nodes and the verification overhead per task

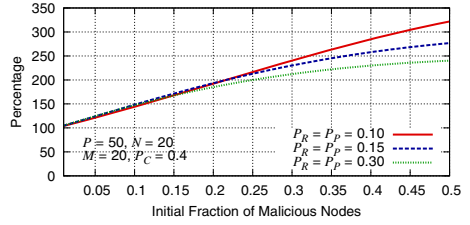


Fig. 8. Expected number of rounds to complete a task as a function of the initial fraction of malicious nodes and the verification overhead per task

$1 - (1 - P_{Z_i})^{(j+1)} = 1 - (1 - P_{Z_i})^2$. The actual number of subtasks that must be discarded depends on the execution trace of the process; in the best case, only the current execution must be repeated. In the worst case, the first subtask was already run on a malicious participant and, due to the sequential property of the function, the entire task execution must be restarted. Given that subtasks are assigned to participants uniformly at random, any of these events is equally likely. In state $s_{2,1}$ this means that either the current or the current plus the already completed subtask must be discarded with probability $\frac{1}{2}(1 - (1 - P_{Z_i})^2)$ each. The former case is represented by the state $s_{3,1}$ the latter by the state $s_{3,0}$.

Based on the above observations, we derive an (recursive) expression for the expected number of required state transitions (i.e., rounds) to reach an absorbing state (i.e., to terminate the task execution) from the current state $s_{2,1}$. Let $P_{X_{i,j}} = 1 - (1 - P_{Z_i})^{(j+1)}$ be the probability that at least one subtask was run on a malicious participant, then $E[Y_{2,1}]$ can be computed as:

$$E[Y_{2,1}] = 1 + \frac{P_{X_{2,1}}}{1+1} \sum_{k=0}^1 E[Y_{3,k}] + (1 - P_{X_{2,1}})E[Y_{3,3}].$$

Generalizing this result to an arbitrary state $s_{i,j}$ yields:

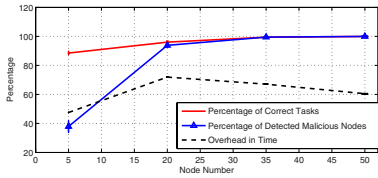
$$E[Y_{i,j}] = \begin{cases} 0, & \text{if } j = M, \\ M - j, & \text{if } i = i^*, \\ 1 + \frac{P_{X_i}}{j+1} \sum_{k=0}^j E[Y_{i+1,k}] + (1 - P_{X_i})E[Y_{i+1,j+1}], & \text{otherwise.} \end{cases}$$

The first case trivially follows from the fact that a task terminates once all M subtasks have been executed and their results been accepted. The second case uses the observation that after i^* rounds all malicious nodes have been eliminated and that the remaining $M - j$ subtask therefore can be computed in $M - j$ rounds. The third case follows from the above discussion on the example of state $s_{2,1}$. Finally, the expected number of rounds to complete an entire task is given by $E[Y_{0,0}]$ and can be computed by means of recursive insertion.

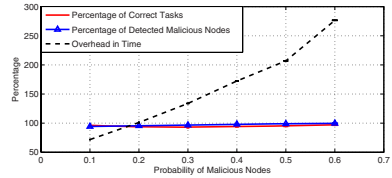
The expected number of rounds to complete a task as a function of the initial fraction of malicious nodes and the verification overhead per task (P_R and P_P) is shown in Figure 8. In this example, the number of tasks and subtasks is 20, the number of participants 50, and initially 10% of the nodes are malicious and cheat with a probability of 40%. We observe that even for a moderate initial fraction of malicious nodes, the actual expected value is significantly lower than the expected upper bound (depicted in Figure 7). That is, to perform the task executions in parallel to the node set purification takes less time than it takes to first identify and remove the malicious nodes and subsequently execute the tasks on a set of honest nodes. We therefore conclude that the naive approach in which one tries to “clear” the node set prior to the actual task executions (by running preliminary extra-computations) is *only* reasonable if the number of subtasks M is comparatively high. Note that the number of subtasks in which a task can be split depends on the structure of the task and on the induced overhead for communication and synchronization per subtask. In order to account for the impact of a non-equal number of subtasks per task as well as for other realistic conditions such as a (possibly) imperfect (pseudo-random) permutation of task assignments, we further evaluate the performance of our scheme by means of extensive simulations.

4.1 Evaluation Results

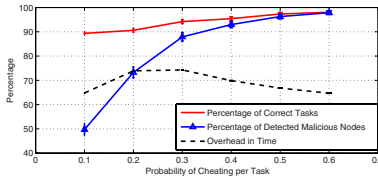
We implemented a C-based simulator to evaluate the performance of our proposal in realistic settings and with respect to various parameters. Our simulator is sequential



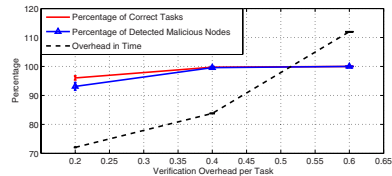
(a) Effect of the Number of Participants.



(b) Effect of the Probability of Malicious Participants.



(c) Effect of the Probability of Cheating per Task.



(d) Effect of the Probability of Ringers/Redundancy per Task.

Fig. 9. Performance of our scheme with respect to various parameters. Each data point in our plots is averaged over 1000 measurements. We show the corresponding 95% confidence intervals. Our results suggest that our scheme can ensure that a substantial percentage of the tasks had been correctly executed while incurring an acceptable overhead in time (typically less than twice as much as the time needed to re-run all the N tasks on trusted nodes).

and round-based. It takes as input the number of tasks and subtasks (default value is 20, respectively), the number of participants (22 by default), the number of malicious nodes in the system (10% of the nodes are malicious by default), the probability of cheating per malicious node (default value = 40%) and the probability of inserting security checks within tasks (default value for the fraction of ringers = 10% and selective redundancy fraction = 10%). We chose a modest number of participants and tasks in our simulations (< 50) to better emulate realistic settings. Note, however, that the performance of our scheme considerably improves as the numbers of participants and tasks increase (Equation 2).

Throughout our simulations, we consider hybrid schemes where the supervisor equally relies on the use of embedded ringers and selective subtask redundancy. As discussed previously, such schemes are likely to be less resilient to malicious behavior when compared to the solutions where the supervisor solely makes use of ringers. We argue, therefore, that our findings presented thereafter correspond to a worst-case scenario, where the supervisor has only a limited number of ringer candidates. Conforming with our analysis in Section 3 we assume perfect collusion between malicious participants; whenever two or more malicious participants run the same subtask, they will all report the same incorrect result to the supervisor in an attempt to decrease the probability that they get caught. In our simulations, the various tasks are broken into smaller subroutines according to the control flow structure of their function. To better analyze synchronization costs in practical settings, we randomly vary the length of each subtask. The supervisor then permutes and assigns these subtasks among participants as shown in Figure 1. To ensure the correctness of the executing tasks, we adopt the recovery mechanism described in Section 4.

Our results² (Figure 9) confirm the analysis that we conducted in the previous sections; our proposed scheme provides a practical and robust tradeoff between the detection rate and the overhead in time with respect to the assurance level in the correctness of the tasks. In fact, even in scenarios featuring 20% colluding malicious nodes, our scheme can achieve a satisfactorily large detection rate and ensures that a substantial percentage of the tasks had been correctly executed ($> 90\%$) while incurring an overhead in time³ that is typically less than twice as much as the time needed to execute all the N tasks on trusted nodes. Needless to mention, as the number of malicious nodes in the network increases, a larger fraction of ringers and/or redundancy is needed to prevent possible misbehavior in our scheme. In turn, this increases the number of required re-runs per task and subsequently the time required to complete the tasks' execution to ensure a satisfying level of confidence in the results (Figure 9(b)).

5 Discussion

Efficient recovery mechanisms from possible misbehavior within the remote execution constitute an important and orthogonal problem to securing distributed computations.

² In our plots, "Overhead in Time" refers to the *additional* overhead incurred by re-running a subset of the subtasks (e.g., an overhead of 50% means that our scheme results in 50% increase in execution time when compared to the time required to run a task in trusted settings).

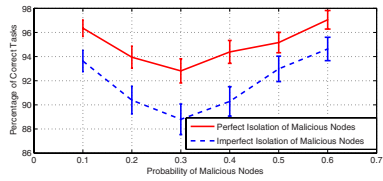
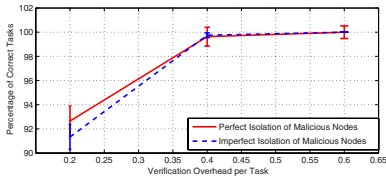
³ This equally includes the synchronization overhead among nodes in each round.

This problem is further aggravated when dealing with sequential computations; a single erroneous undetected subtask affects the correctness of the entire task. In the previous section, we attended to this “hitch” and we presented a framework that facilitates efficient recovery from erroneous computations.

Non-Uniform Sampling Distributions: Throughout our analysis, we considered the case where the supervisor checks a sample of the computations following a uniform distribution. Uniform distributions proved to provide a strong form of probabilistic security because they limit the number of viable countermeasures that untrusted hosts can adopt (these hosts equally have to adopt a uniform cheating strategy to increase their advantage in the network). However, uniform sampling distributions might be less optimal with respect to the efficiency of recovering from erroneous computations in sequential tasks; indeed, when relying on such distributions, the probability to catch an erroneous mistake in subtask # 1 is equal to the probability of catching an error in subtask # M . Although this comes at the benefit of a superior overall detection rate, this seems to be suboptimal in practice. In typical cases, the supervisor is more interested in catching misbehavior *earlier* during the execution since this implies less recovery overhead (an undetected error in subtask # 1 renders all subsequent computations erroneous). This suggests that biased sampling distributions – in which the supervisor checks more often the preliminary subtasks – are likely to require less re-runs to ensure the correctness of the tasks, thus boosting the recovery performance. Biased distributions come, however, at the cost of reduced detection rate; malicious participants can cheat more often in the final computations without being detected.

The optimal tradeoff between the performance in detecting malicious participants and the efficiency in recovering from malicious behavior emerges as an interesting research problem. For instance, one alternative would be to combine the use of both biased and uniform sampling distributions; this can be achieved by relying on biased checks within each task and permuting the tasks among participants such that these checks are almost uniform amongst the subtasks that each participant executes. Given this scheme, a participant is likely to expect an (almost) uniform sample-checking whereas, from the supervisor’s point of view, early subtasks are checked more often than later ones. This solution requires, however, that the execution time of the tasks in the system equally follows a biased distribution in time.

Impact of Imperfect Node Isolation: So far, our analysis was based on the assumption that, once identified, malicious nodes can be completely excluded from the subsequent computation rounds. This corresponds to a closed network system where each entity can be uniquely identifiable. However, participants might be able to operate under several identities (Sybil attack [21]) or to re-enter the system with a new identity (i.e., white-washing), which renders this assumption rather unrealistic in typical settings. In such “open” systems, the supervisor might not be able to fully isolate malicious participants; the overall fraction of malicious nodes tends to remain, to a large extent, constant in this case. Here, as opposed to a (partially) closed system, sample checking solely protects past computations performed by the participants and does not improve the conditions for the subsequently executed subtasks. Nevertheless, even in this case, our results show (refer to Figure 10) that our scheme achieves a high level of confidence in the computed



(a) Effect of the Probability of Ringers/Redundancy per Task. (b) Effect of the Probability of Malicious Participants.

Fig. 10. Impact of Imperfect Isolation of Malicious Participants on the performance of our scheme. Our findings are derived from the simulation setup described in Section 4.1. Our results suggest that our proposed scheme achieves high confidence in the correctness of the tasks even in scenarios where perfect isolation of malicious participants might not be possible.

results. This indicates that although isolating malicious nodes is clearly beneficial, it is not a requirement to achieve reliable results in our scheme; due to the high performance of our scheme in detecting malicious behavior, the supervisor can ensure the correctness of its executing tasks by re-running a modest subset of the tasks, as described in Section 4. Note that the performance of the entire system can be further ameliorated through the use of reputation-based approaches (e.g., [15], [16], [17], [18]); in this case, each participant can be associated with a reputation value that indicates how trustworthy it is. Such an approach enables the supervisor to start with better knowledge of the participants’ credibility (i.e., the initial probability of malicious participants is slightly lower when compared to open systems) and therefore bridges the gap between the aforementioned extremes: closed systems in which malicious participants can be fully isolated and open systems, where malicious nodes cannot be isolated from the system.

6 Conclusion

While there are several proposals that address the security of distributed non-sequential functions, the literature includes very few proposals for securing remote sequential computations. In this paper, we address this problem and we show that by permuting sequential tasks among several participants, efficient probabilistic measures can be used to secure the remote execution of tasks. More specifically, we demonstrate that our proposal enables a remote supervisor to selectively embed indistinguishable security checks within the sequential computations and we show that the resulting scheme facilitates the detection of individual and colluding malicious participants that cheat in a subset of the computations. We further discussed mechanisms that facilitate recovery from possible chaining of errors within the ongoing remote computations. Our findings indicate that by capitalizing on the high detection rate of our scheme to identify malicious participants, a satisfactorily modest number of re-runs per task can ensure high confidence levels in the correctness of all the tasks in the system, thus bounding the impact of malicious behavior.

References

1. SETI@home, <http://setiathome.ssl.berkeley.edu/>
2. Distributed.Net, <http://distributed.net/>
3. The Great Internet Mersenne Prime Search, <http://www.mersenne.org/prime.htm>
4. Golle, P., Mironov, I.: Uncheatable Distributed Computations. In: Proceedings of the RSA Conference (2001)
5. Szajda, D., Lawson, B., Owen, J.: Hardening Functions for Large Scale Distributed Computations. In: Proceedings of the IEEE Symposium on Security and Privacy (2003)
6. Wasserman, H., Blum, M.: Software Reliability via Runtime Result-Checking. *Journal of the ACM* (1997)
7. Golle, P., Stubblebine, S.: Secure Distributed Computing in a Commercial Environment. In: Proceedings of the International Conference on Financial Cryptography (2001)
8. Sander, T., Tschudin, C.F.: Protecting Mobile Agents Against Malicious Hosts. *Mobile Agent Security* (1998)
9. Vigna, G.: Protecting Mobile Agents Through Tracing. In: Proceedings of the ECOOP Workshop on Mobile Object Systems (1997)
10. Du, W., Jia, J., Mangal, M., Murugesan, M.: Uncheatable Grid Computing. In: Proceedings of ICDCS (2004)
11. Goodrich, M.T.: Pipelined Algorithms to Detect Cheating in Long-Term Grid Computations. *Theoretical Computer Science* (2008)
12. Yang, S., Butt, A.R., Hu, Y., Midkiff, S.P.: Lightweight Monitoring of the Progress of Remotely Executing Computations. In: Proceedings of the International Workshop on Languages and Compilers for Parallel Computing (2007)
13. Jin, H., Lotspiech, J.: Forensic Analysis for Tamper Resistant Software. In: Proceedings of ISSRE (2003)
14. Linn, C., Debray, S.: Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In: Proceedings of CCS (2003)
15. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: WWW 2003 (2003)
16. Damiani, E., Paraboschi, S., Samarati, P., Violante, F.: A Reputation-based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In: Proceedings of the ACM Conference on Computer and Communications Security (2005)
17. Dimitriou, T., Karame, G., Christou, I.: SuperTrust: A Secure and Efficient Framework for Handling Trust in Super Peer Networks. In: Proceedings of ACM PODC (2007)
18. Karame, G., Christou, I., Dimitriou, T.: A Secure Hybrid Reputation Management System for Super-Peer Networks. In: Proceedings of IEEE CCNC (2008)
19. Baron, M.: Probability and Statistics for Computer Science. Chapman & Hall/CRC, Boca Raton (2007)
20. Haeberlen, A., Kuznetsov, P., Druschel, P.: PeerReview: Practical Accountability for Distributed System. In: Proceedings of ACM SOSP (2007)
21. Douceur, J.: The Sybil Attack. In: Proceedings of the IPTPS Workshop, Cambridge, MA, USA (2002)

Architecture- and OS-Independent Binary-Level Dynamic Test Generation

Gen Li, Kai Lu, Ying Zhang, Xicheng Lu, and Wei Zhang

School of Computer, National University of Defence Technology, ChangSha, China
superligen@gmail.com, {kailu,zhangying,xichenglu,wz}@nudt.edu.cn

Abstract. Dynamic test generation approach consists of executing a program while gathering symbolic constraints on inputs from predicates encountered in branch statements, and of using a constraint solver to infer new program inputs from previous constraints in order to steer next executions towards new program paths. Variants of this technique have recently been adopted in finding security vulnerabilities in binary level software. However, such existing approaches and tools are not retargetable: on the one hand, they can only find vulnerabilities in the binaries for a specific ISA; on the other hand, they can only find vulnerabilities over a specific OS because the execution trace is totally OS-dependently recorded in these tools. This paper presents a new dynamic test generation technique and a tool, ReTBLDTG, short for *ReTargetable Binary-Level Dynamic Test Generation*, that implements this technique. Unlike other such techniques, ReTBLDTG can deal with binaries for any ISAs over any OSes. ReTBLDTG is based on the whole system virtual machine that provides OS-independent and fast *concrete execution* of the target program. And which thread the executing instruction belongs to is OS-independently identified by analyzing the registers' value and hardware events over the virtual machine. Thus, the execution trace is recorded, without knowing the internal structure of the guest OS. At the same time, ReTBLDTG defines a *Meta Instruction Set Architecture* (MetaISA); ReTBLDTG maps the execution information, which is collected during the binary source code execution, to MetaISA; and symbolic execution, constraint collection and constraint solver operates on MetaISA, thus making these tasks ISA-independent. We have implemented our ReTBLDTG, retargeted it to 32-bit x86, PowerPC and Sparc ISAs, and used it to automatically find the six known bugs in the six benchmarks over Linux and Windows. Our results indicate that our ReTBLDTG can be easily retargeted to any ISA with only a few overheads; and ReTBLDTG can effectively find bugs located deep within large applications over any OS.

1 Introduction

Dynamic test generation approach, like DART [6], EXE [3] and SAGE [7], is becoming increasingly popular to find security vulnerabilities in software. Starting with a fixed input, the approach symbolically executes the program, gathering input constraints from conditional statements encountered along the way.

The collected constraints are then systematically negated and solved with a constraint solver, yielding new inputs that exercise different execution paths in the program. For example, symbolic execution of the conditional statement “if $(x==10)$ then” on the input $x = 0$ generates the constraint $x \neq 10$. Once this constraint is negated and solved, it yields $x = 10$, which gives us a new input that causes the program to follow the then branch of the given conditional statement. This allows us to exercise and test additional code for security bugs, even without specific knowledge of the input format. Furthermore, this approach automatically discovers and tests corner cases where programmers may fail to properly allocate memory or manipulate buffers, leading to security vulnerabilities.

More and more research institutes and groups use this approach to find security vulnerabilities in the pre-release software, which is usually shipped in binary code, after it has been heavily tested using a combination of code review, manual and random testing, dynamic tools and static analysis, because it finds security vulnerabilities without generating false alarms and requires no domain knowledge.

However, existing such tools are not retargetable. On the one hand, they can only find vulnerabilities in the binaries for a specific ISA, due to specific architecture details of different *Instruction Set Architectures* (ISAs).

On the other hand, they can only find vulnerabilities over a specific OS because the execution trace is totally OS-independently recorded in these tools. The execution trace, made up of the instruction flow, of the target program run with the initial input, should be recorded for the following constraint generation and solver. Currently, the execution trace is recorded as the program is executed either by statically injected instrumentation code or with the help of binary instrumentation tools such as Nirvana [2] or Valgrind [12]. However, these instrumentation tools strongly depend on the OS (Operation System), thus make existing dynamic test generation un-retargetable. These instrumentation tools run over the guest OS and call OS-dependent *Application Programming Interfaces* (APIs) to identify the process and its threads of the target program, and monitor its thread switch. Because the process and thread management are totally defined by the OS, the instrumentation tools must know the internal structure of guest OS.

The coupling of binary-level dynamic test generation with specific architecture or OS details creates an interoperability problem that hinders the wide adoption of binary-level dynamic test generation. To adopt this approach to find security vulnerabilities for any other ISAs or over any other OSes, one has to develop another separate tool for the specific ISA or OS.

This paper presents a new binary-level dynamic test generation technique and a tool, ReTBLDTG, short for *ReTargetable Binary-Level Dynamic Test Generation*, that implements this technique. Unlike other dynamic test generation techniques that operate only on binaries for a specific ISA, ReTBLDTG can process binaries for any ISAs over any OSes and dynamically generates new inputs that exercise different control paths in the program, which may lead to security vulnerabilities.

To mask the difference of the CPU ISAs, ReTBLDTG defines a *Meta Instruction Set Architecture* (MetaISA). When working, ReTBLDTG maps the execution information, collected during the execution of binary source code, to MetaISA. And symbolic execution, constraint collection and constraint solver operate on the code in our MetaISA, thus making the three processes ISA-independent. As shown in Section 2, ReTBLDTG consists of 208KLOC and these three processes represent over 94% of our code base. Thanks to MetaISA, ReTBLDTG is retargetable with only a few overheads. To port ReTBLDTG to a new CPU platform, we only need to implement a new decoder and an ISA mapper for it.

Because symbolic execution, constraint collection and constraint solver operate on the code in our MetaISA, the following key issues must be considered when MetaISA is designed:

- The MetaISA should be as simple and uniform as possible in order to facilitate the following three processes. This requires all meta instructions should be arithmetic instructions. The conditional instructions, such as `cmp`, should be transformed to the change to flag bit, including `ZF`, `OF`, and `CF`, and so on; the branch instructions, such as `Jnz`, should be transformed to the selection of PC based on some registers' value; the complex instructions, such as `bsf` (Bit Scan Forward) that searches the source operand for the least significant set bit, and `rep movsd` that copies data from source to destination until `ecx == 0` from x-86 ISA, should be expressed by the combination of simple arithmetic instructions. Thus, the constraints expressed in simple meta arithmetic instructions can be mapped to the SMT solver smoothly. Additionally, this can also simplify the symbolic execution.
- Each instruction operation of the MetaISA should be bit-precision. This requires how each bit of each variable of the left-hand side (LHS) of a meta instruction is computed from every bit of the right-hand side (RHS) must be precisely expressed. This is because the SMT solver, used to generate a new input excising to a different control path based on the gathering constraints, adopts *bit-vector* theory that demands all constraints expressed as bit-precision .
- The design for MetaISA should consider its effect on performance and memory consuming of ReTBLDTG. In 32-bit x86 ISA, for example, most instructions operate on 32-bit data and only a few, such as `mul`, `div` and `mod`, generated 64-bit medium data. When 32-bit x86 ISA is mapped to MetaISA, it is easy to map all operands to 64-bit registers. However, when ReTBLDTG deals with very large real applications with millions of instructions, the total memory requirement of symbolic execution, constraint collection and constraint solver would be huge. However, if all 32-bit x86 instructions are mapped to meta instructions operating on 32-bit registers, the total memory requirement can be reduced to nearly half, and improve the efficiency of the SMT solver.

To mask the difference of the OSES, the execution trace of the target program should be under the OSES or over the naked CPU. Otherwise, the identification of

the process and its thread must call the OS API. Fortunately, running the target program on the whole system simulator is a good choice to address the above problem: on the one hand, the registers and hardware events can be monitored over the whole system virtual machine, without any help of OSes; on the other hand, the whole system virtual machine can execute the target program over the guest OS with quite fast and acceptable speed compared with the time-consuming constraint collection and solver.

ReTBLDTG is based on the whole system virtual machine Simics but is not dependent on it. The whole system virtual machine provides OS-independent and fast *concrete execution* of the target program. And which thread the executing instruction belongs to is OS-independently identified by analyzing the registers' value and hardware events over the virtual machine. Thus, the execution trace is recorded, without knowing the internal structure of guest OS.

The main contributions of this paper are as follows.

- We design a meta ISA;
- We present a method to online identify the process;
- We present, for the first time to the best of our knowledge, a method to online identify the thread;
- We build, for the first time, a new binary-level dynamic test generation technique and a tool, ReTBLDTG, that can find bugs from binaries for any ISA over any OSes, based on a whole system virtual machine;
- We have retargeted ReTBLDTG to 32-bit x86, PowerPC and Sparc ISAs by now;
- We have retargeted ReTBLDTG for the Linux and Windows binaries by now;
- Our ReTBLDTG efficiently found the bugs from the Linux and Windows binaries for 32-bit x86, PowerPC and Sparc ISAs.

The rest of this paper is organized as follows. The ReTBLDTG system architecture is given in Section 2. Section 3 identifies the process and its thread of the running target program based on the execution of virtual machine. In Section 4, we show how MetaISA is designed in order to make ReTBLDTG architecture-independent. Our experiments and performance evaluation appear in Section 5. The related work is discussed in Section 6 and we conclude in Section 7.

2 The ReTBLDTG System Architecture

As shown in Figure 1, ReTBLDTG is built around four levels of abstraction to make it architecture- and OS-independent. Presently, ReTBLDTG consists of 208KLOC with 0.6% in the Virtual Execution Layer (VEL), 5.0% in the Process/Thread Identification Layer (PTIL), 1% in the MetaISA Layer (ML) and 94% in the Constraint Analysis Layer (CAL). We describe each layer only briefly in the rest of this section and focus mostly on introducing how PTIL as well as VEL make CAL OS-independent in Section 3, and how ML makes CAL ISA-independent in Section 4.

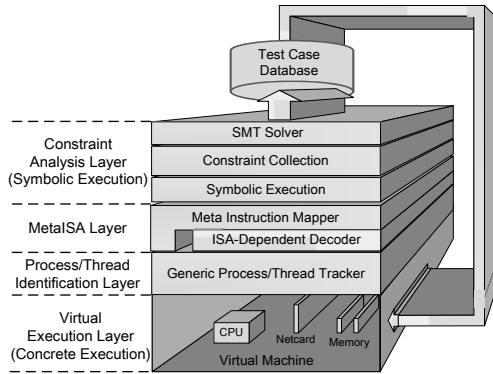


Fig. 1. ReTBLDTG system architecture

2.1 Virtual Execution Layer

VEL is essentially a virtual machine (VM) or a whole system emulator like VMware, Simics or QEMU. Presently, ReTBLDTG is based on Simics but is not dependent on it. VEL provides OS-independent and fast *concrete execution* of an application.

As a VM, ReTBLDTG is equipped with the capability of analyzing applications that are time-sensitive or protected by anti-debugging techniques. ReTBLDTG can freeze the entire system, including the clock when performing time-consuming tasks like points-to analysis at a higher layer. As a result, the OS and the application are not even aware of the elapse of the time. Furthermore, all test cases can be restarted from exactly the same system state.

2.2 Process/Thread Identification Layer

PTIL distinguishes different processes and segments the instructions flowing through the CPU in the same process into different instruction sequences belonging to different threads. This is necessary since different threads in a process should have their own sets of registers associated with them.

The principle behind PTIL is based on some hardware events, independent of the OS. For example, in x86 ISA, process switching is recognized by listening to the CR3-Changed-Event as in [8]. Thread identification appears to be novel (to the best of our knowledge). Thread starting or stopping is recognized by identifying a thread using its stack pointer (ESP) and by monitoring the CPU privilege level transitions between Ring0 and Ring3. These ideas generalize to other architectures.

2.3 The Meta Instruction Set Architecture

One key motivation for introducing a MetaISA is to make CAL ISA-independent. Another is to facilitate our symbolic execution, constraint collection and constraint solver on a simple and uniform MetaISA. The entire CAL layer represents

over 94% of our code base. Thanks to MetaISA, CAL is now retargetable. To port ReTBLDTG to a new CPU platform, we only need to implement a new decoder and an ISA mapper for it.

2.4 Constraint Analysis Layer

As shown in Figure 1, CAL is responsible for performing symbolic execution, constraint collection and constraint solver. This layer focuses on a process being analyzed and ignores the instruction sequences from the other processes and the OS kernel, which is made possible by the PTIL layer below. This layer represents over 94% of our code base and performs the most time-consuming tasks of our ReTBLDTG as evaluated in Section 5. Our CAL is like the of SAGE. Refer to [7] for more details.

3 Making CAL OS-Independent

The VEL and PITL work together to make CAL OS-independent. The VEL in ReTBLDTG has two tasks that: 1. VEL fast executes the instruction flow, mixed by guest operating system and destination application; 2. it also submits the specific hardware events like *Page Table Switch* (In x86, it is CR3-Changed-Event) and *CPU Privilege Switch* events and allows PTIL to listen to. Based on these hardware events, PTIL splits the single instruction sequence, flowing CPU, into instruction sequences belonging to different threads of the target process.

It is a meticulous consideration that ReTBLDTG adopts a whole system virtual machine based online approach, instead of an instrumentation as existing dynamic test generation systems. Thus, ReTBLDTG has more flexibility, independent on any specific OS. As discussed before, the instrumentation tools, like Valgrind and iDNA, strongly depend on the OS API. ReTBLDTG is based on the virtual machine and can transparently monitor the registers and hardware events of the virtual machine. Thus, ReTBLDTG records the execution trace by analyzing the CPU behavior, without knowing the internal structure of guest OS. Recording the execution trace will not be affected by the operating system protection or application self-protection. The instrumentation tools, like iDNA [2] and Valgrind, will be disturbed by the software protections, such as Anti-Debug. These protections should not be removed during systemic test because these protections are also part of the software under test. VEL watches the guest OS execution in the view of CPU. Thus, the target running program cannot feel the existence of ReTBLDTG. Therefore, VEL can not be disturbed by software self-protection. ReTBLDTG can effectively process the time-sensitive applications, particularly network applications. For time-sensitive applications, like network applications, the time-pause caused by CAL may lead a timeout for receiving/sending a packet. When ReTBLDTG does time-consuming task, such as constraint collection and solver, ReTBLDTG freeze the whole system clock, without the guest OS and target program aware of it. But the existing dynamic test generation tools, based on instrumentation tools, can not correctly process time-sensitive applications.

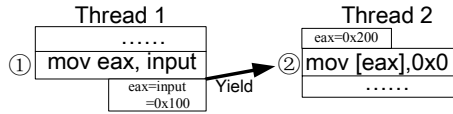


Fig. 2. A case showing that the thread switch interferes with the symbolic execution

The PTIL, as shown in Figure 1, precisely OS-independently tracks the target process and its threads. It dynamically filters the instructions flowing through CPU and segments them into different instruction sequences belonging to different threads of processes, and ReTBLDTG binds each instance of *thread trackers* to track the instruction sequence for each thread.

We must accurately identify each thread of the target process because each thread has its own register space. As shown in Figure 2, when thread 1 is executing instruction 1, `eax=input=0x100`. Assuming that thread 1 switches to thread 2 before executing its next instruction. The concrete execution of the guest OS has switched to the register context of thread 2, with `eax` updated to `0x200`, before thread 2 runs. Because the symbolic execution of CAL is independent on the concrete execution, the real register status, maintained by virtual machine, must be kept for the following CAL analysis. Otherwise, when the instruction `mem[eax] = 0` is executed in thread 2, the symbolic execution will be diverged from the concrete execution.

Identifying target process/thread. Segmenting the single instruction sequence flowing through CPU into different instruction sequences, belonging to different threads of each process, can accurately track each process and its threads.

Identifying the processes. Nowadays, according to the implementation of the mainstream OSes, every process has its own page table, pointed by CR3 register and used to isolate virtual address resource by MMU. When the CR3-Changed-Event happens, we can identify that the process must be switched. In Figure 3, the CR3 is changed to `0x1000`, ReTBLDTG looks at the new value of CR3 as the PID of the switched-in process.

Identifying the threads. We can find that every process-switch event must happen in the CPU privilege level of Ring0. After the privilege level is dropped to Ring3, the instruction sequence to be executed must belong to one of the current process's threads. Generally, the *stack pointer* (ESP) can be used to effectively distinguish and identify the threads of the same process because each thread has its own stack. Different from the CR3 register, ESP is changed along with the execution of the current thread. ReTBLDTG records the value of ESP into the *thread ID* (TID) list when the CPU privilege level raised to Ring0; and ReTBLDTG checks whether the current ESP has been recorded in the TID list when the CPU privilege level drops to Ring3 in the same process. If yes, ReTBLDTG appends the instruction sequence to be executed before next switching to the execution trace of the found thread; otherwise, ReTBLDTG identifies a new thread.

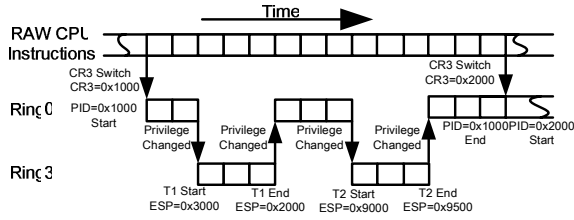


Fig. 3. Tracking generic process/thread

Figure 3 describes how to track generic threads. When the CPU privilege goes to Ring3 in the process 0x1000 for the first time, ReTBLDTG identifies that T1 is the current thread. The instruction sequence from T1 Start to T1 End belongs to T1. After T1 End, the CPU privilege rises to Ring0. Before entering Ring0, ReTBLDTG reads 0x2000 from the current stack pointer (ESP), and makes the value map to T1: 0x1000→T1. When CPU returns to Ring 3 in the same process, ReTBLDTG examines whether the value of ESP has been mapped. In this case, the current ESP equal to 0x9000, there is no thread mapped to 0x9000. ReTBLDTG justifies the current thread is not T1, and identifies it as T2.

4 Making CAL ISA-Independent

One key motivation for introducing a MetaISA is to make CAL ISA-independent. Another is to facilitate our symbolic execution, constraint collection and constraint solver on a simple and uniform MetaISA. The entire CAL layer represents over 94% of our code base. Thanks to MetaISA, CAL is now retargetable. To port ReTBLDTG to a new CPU platform, we only need to implement a new decoder and an ISA mapper for it.

As shown in Table 1, MetaISA adopts the little endian format and defines four types of registers with 128 non-aliased and interchangeable registers in each category. Our meta instructions are specified using the Semantic Specification Language (SSL) [13].

To facilitate the constraint collection, MetaISA has the following features.

Every meta instruction is a bit-precision assignment and the bit-width of RHS and LHS of a meta instruction is equal. In dynamic test generation systems, the conditional constraints are first collected from the symbolic execution, and then the constraints in meta instructions are transformed into logic conditional constraints and putted into the SMT solver. Therefore, the constraints in meta assignment instructions can be smoothly transformed into the logic == constraints. At the same time, bit-precision assignment instructions can precisely depicts how the left variable is computed from the right variables. Thus, the derived logic constraints can bit-precisely show how the collected constraints are affected by the input or medium variables. Additionally, SMT solver, used to generate a new input excising to a different control path based on the

Table 1. Meta Instruction Set Architecture

Endian	Little endian
32bits Regs	R_0, R_1, \dots, R_{100}
64bits Regs	A_0, A_1, \dots, A_{100}
80bits Float Regs	F_0, F_1, \dots, F_{100}
128bits Float Regs	X_0, X_1, \dots, X_{100}
Instructions i	$Reg = f(Reg, mem[g(Reg)])$ $mem[g_1(Reg)] = f(Reg, mem[g_2(Reg)])$
Operands v	Reg, Imm, $mem_{8,16,32,64}[g(Reg)]$
Operations Δ_t	exact, sigext, zeroext, (...) ? (...):(...)
Δ_b	+, -, *, /, \oplus , $\&$, $-$, \ll , \gg , $=$, $<$, \leq
Δ_s	\neg

gathered constraints, adopts *bit-vector* theory which demands all constraints expressed as bit-precision.

Every flag bit is defined as a register. Because input constraints are gathered from conditional statements encountered along the way, the branch instruction and the condition instruction pair should be recognized. The existing binary-level dynamic test generation tools, such as SAGE [7], search the most recent conditional instruction when meeting a branch instruction, in order to find the pair. However, compilation optimization of prefetching might insert a block of instructions between the pair [11], which makes it not an easy thing to recognize the branch instruction and the condition instruction pair. In our MetaISA, we define each flag bit, including ZF (Zero Flag), OF (Overflow Flag), CF (Carry Flag), and so on, as a register, called FLAG REG, and assign conditional constraints to corresponding FLAG REGs. Thus, how conditional instructions change the flag bit is kept in these registers. Searching the FLAG REGs we can get how the conditional constraints are affected from input or medium variables when meeting a branch instruction.

All general-purpose registers are non-aliased. If some register bits have alias, which causes two register identifiers point to the same register bits, it brings difficulty to the analysis, collection and solver of the conditional constraints. Therefore, it is a smart choice that the registers in the MetaISA are independent without any alias. And skillful work should be done for the CPU ISA that has aliased register bits.

All registers have the same width. Some source binary instructions, such as mul, div and mod, may involve registers of double-size. For example, the registers, used to hold the result of a 32-bit integer multiplied by a 32-bit integer, is of 64-bit width. And in 32-bit x86 ISA, the lower 32 bits of the result are kept in the destination register as the multiplication result; the higher 32 bits is used to compute the *Carry Flag* (CF) and *Overflow Flag*(OF), according to whether the multiplication operation is overflow or not. As discussed before, registers of the MetaISA with large width might bring about bad performance for SMT solver and huge memory requirement. Therefore, in our MetaISA, all instructions operate on the data with

the same width and all registers have the same bit-width. For the instructions involved with double-size registers, we use `exact`, `sigext` or `zeroext` operation to extend the intermediate results to necessary size.

Table 2 lists the map of `imul` and `idiv`, the representative instructions involved with the results of double-size, to our MetaISA.

Table 2. Mapping the instructions `imul` and `idiv` to our MetaISA

Ins.	Map to our MetaISA
<code>imul ebx,edi</code>	<pre>tmp1 := ebx ebx := edi *! tmp1 CF := (((zeroext(edi,32,64) *! zeroext(tmp1,32,64)) == zeroext(ebx,32,64)) ? 0x0 : 0x1)[31:0] & 0x1 OF := (((zeroext(edi,32,64) *! zeroext(tmp1,32,64)) == zeroext(ebx,32,64)) ? 0x0 : 0x1)[31:0] & 0x1</pre>
<code>idiv ecx</code>	<pre>tmp1 := eax eax := (((zeroext(edx,32,64) << 0x20) zeroext(tmp1,32,64)) /! zeroext(ecx,32,64))[31:0] edx := (((zeroext(edx,32,64) << 0x20) zeroext(tmp1,32,64)) %! zeroext(ecx,32,64))[31:0]</pre>

5 Experimental Evaluation and Results

We have implemented our ReTBLDTG system in 208KLOC (Kilo Lines Of Code). And in order to demonstrate the effectiveness of our new approach on decoupling binary-level dynamic test generation from specific architecture details, we have retargeted ReTBLDTG to three different architectures, including 32-bit x86 ISA, PowerPC ISA and Sparc ISA by now. 32-bit x86 ISA falls into CISC; while PowerPC ISA and Sparc ISA belong to RISC.

We first demonstrate the importance of our ReTBLDTG on decoupling binary-level dynamic test generation from specific architecture details. Table 3 shows the workload for retargeting our ReTBLDTG to a new ISA, quantified with the number of LOC of the MetaISA Layer for each ISA. To retarget our ReTBLDTG to 32-bit x86 ISA, 2483 LOC are needed; to PowerPC ISA, 647 LOC are needed; and to Sparc ISA, 835 LOC are needed. Because 32-bit x86 ISA is CISC, a lot of work is needed for the complex instructions. At the same time, compared with the other two ISAs, 32-bit x86 ISA has more instructions. PowerPC ISA and Sparc ISA both belong to RISC. Sparc ISA has more instructions, and retargeting our ReTBLDTG to Sparc ISA needs more work.

The column *% Retarget Overheads* equals $\#LOC_{Retargeted} / \#LOC_{ReTBLDTG} \times 100\%$, where $\#LOC_{Retargeted}$ is the number of LOC for retargeting our ReTBLDTG to a ISA and $\#LOC_{ReTBLDTG} \times 100\%$ is the total number of LOC for building ReTBLDTG, nearly equal to 208K. This column demonstrates the easiness to retarget our binary-level dynamic test generation tool, ReTBLDTG, to a new ISA.

Table 3. Overheads for retargeting our ReTBLDTG to 32-bit x86 ISA, PowerPC ISA and Sparc ISA

ISA	#LOC	Retarget Overheads
32-bit X86	2483	1.19%
PowerPC	647	0.30%
Sparc	835	0.40%

Only 1.19% of the system has to be rewritten when our system is retargeted to the 32-bit x86 ISA, the most complex ISA of them. However, without the MetaISA, nearly the whole system has to be revised or rewritten. Therefore, our ReTBLDTG can be easily retargeted to any ISA with only a few overheads.

We then demonstrate the effectiveness of our new approach for hunting fatal bugs in benchmark and real-application binaries (without knowing their symbol tables). They are binaries for x86, PowerPC and Sparc ISA respectively and tested over Linux, Windows Vista and Windows Vista.

We show that ReTBLDTG can find 6 classic known bugs and we analyze the performance of ReTBLDTG. All experiments are carried out on an Intel 3.0 GHZ E8400 host PC running 32-bit Windows Vista with 4GB RAM. The VEL (i.e. virtual machine layer) of ReTBLDTG is an essentially wrapper for Simics 3.0.31, a high performance full-system simulator.

Table 4 shows the 6 benchmarks, Apache1, Apache2, OpenSER, MADWiFi, ANI and OldWINS. They are known to have one bug each. The first four small benchmarks are selected from the Verisec Security Benchmark suite [9] representing four different common scenarios causing buffer overflow errors. The binaries of Apache1 and Apache2 for PowerPC ISA are tested over Linux2.6; and the binaries of OpenSER and MADWiFi for Sparc ISA are tested over Linux2.6. The last two are real applications with one known security bug each, MS07-017 for the animated icons (ANI) parser in user32.dll of Windows Vista and MS04-045 in WINS Service in Windows 2000. The binaries of these two applications for 32-bit x86 ISA are tested over Windows Vista and Windows 2000, respectively.

ReTBLDTG has succeeded in finding all 6 known bugs in the six benchmarks. The results show that our ReTBLDTG can effectively find bugs for any ISAs over any OSes.

Table 5 gives the performance data for these applications. The first two rows show the generated and executed test cases. Only 3 to 1487 test cases are executed to find all these bugs. The third row gives the time for our ReTBLDTG

Table 4. Benchmarks. IoF stands for Integer Overflow and BoF stands for Buffer Overflow.

	Benchmark	ISA	OS	Bug Type
Verisec Security Benchmark Suite	Apache1: Apache-CVE-200-4 0940 (Full_Ptr_Bad)	PPC	Linux2.6	BoF by an Infinite Loop
	Apache2: Apache-CVE-2006-3747 Iter2 prefixLong_ptr_bad	PPC	Linux2.6	Off-by-One
	OpenSER: OpenSER-CVE-2006-6749 (Complete_Bad)	Sparc	Linux2.6	Lack of Bounds Checking
	MADWiFi: MADWiFi-CVE-2006-6332e (Ncode_le)	Sparc	Linux2.6	Unchecked Bounds in sprintf
Large Real Apps	ANI: User32.dll-MS-07-017	X86	Windows Vista	Failure to Validate Parameter
	OldWINS: WINS with MS04-045	X86	Windows 2000	Using Input as Pointer

Table 5. Performance results of the benchmarks

	Verisec Security Benchmark				Large Real Apps	
	Apache 1	Apache2	OpenSER	MADWiFi	ANI	OldWINS
#Generated Test Cases	476	192	2125	5	797	807
#Executed Test Cases	131	192	1487	3	82	4
Total Time	4m12s	4m40s	29m47s	3m6s	2h3m29s	2m18s
Symbolic Execution	1m15s	16s	18s	1m35s	1m12s	5s
Concrete Execution	21s	46s	11m8s	<1s	7s	<1s
Constraint Solving	1m35s	2m27s	14m53s	55s	1h26m10s	1m32s
MetaISA Decoding	35s	46s	12s	1s	5m28s	8s
Test Case Database	26s	23s	3m18s	33s	32m32s	14s

to find the bugs. It only takes from 2m18 to 2h3m29s for our ReTBLDTG to find the bugs. The last five rows list the time distribution, which shows that our ReTBLDTG spends most time on symbolic execution, constraint collection and constraint solver. Our ReTBLDTG makes these time-consuming tasks ISA- and OS-independent, thus making dynamic test generation approaches efficiently find bugs from binaries for any ISAs over any OSES with only a few overheads.

6 Related Work

We refrain from discussing a large body of work done on static analysis, program verification, fuzz testing [5], dynamic taint analysis and model checking since good reviews are available in [6,3,14,4,7,12,10]. Instead, we focus mainly on a few techniques that are closely related to our work and that can also be used to test pre-release software in binary code.

Systematic dynamic test generation is becoming increasingly popular because it can find bugs by automatically generating test cases without false positives. DART [6], EXE [3], CUTE [14] and KLEE [4] are a few representatives. By operating on the source code only, these tools do not reason well about bugs that depend on, for example, heap layout at runtime. They represent tainted arrays symbolically (rather than with real addresses) and handle only some limited form of tainted pointers (e.g., scalar pointers only). Our ReTBLDTG has the similar working mechanism as them. But these techniques do not pay their attention to find bugs in binaries, but source code.

SAGE [7] is a dynamic test generation tool that works on Windows binaries. Research group from Berkeley [11] also works hard on finding integer bugs. However, they can only find a specific OS binaries for a specific ISA. And it is a hard and time-consuming work to retarget their techniques to other OSES or ISAs.

7 Conclusion

In this paper, we have introduced the problem of decoupling dynamic test generation from specific architecture and operating system details. We have presented a new binary-level dynamic test generation technique and a tool, ReTBLDTG. ReTBLDTG is based on the whole system virtual machine that provides OS-independent and fast *concrete execution* of the target program. And the execution trace is recorded, even without knowing the internal structure of guest OSES. We also design the MetaISA and map the execution trace to the MetaISA, thus making ReTBLDTG ISA-independent. We have implemented our ReTBLDTG, retargeted it to 32-bit x86, PowerPC, Sparc ISAs and Linux, Windows Vista, Windows Vista OSES, and used it to automatically find the six known bugs in the six benchmarks. Our results indicate that our ReTBLDTG can be easily retargeted to any ISA with only a few overheads and operate on any OSES; and ReTBLDTG can effectively expose bugs located deep within large applications for any ISAs over any OSES.

References

1. Bacon, D.F., Graham, S.L., Sharp, O.J.: Compiler transformations for high-performance computing. *ACM Comput. Surv.* 26(4), 345–420 (1994)
2. Bhansali, S., Chen, W., de Jong, S., Edwards, A., Murray, R., Drini, M., Mihoka, D., Chau, J.: Framework for instruction-level tracing and analysis of program executions. In: *Proceedings of the 2nd international conference on Virtual execution environments*, vol. 14, pp. 154–163 (2006)
3. Cadar, C., Ganesh, V., Pawlowski, P.M., Dill, D.L., Engler, D.R.: Exe: automatically generating inputs of death. In: *CCS 2006: Proceedings of the 13th ACM conference on Computer and communications security*, pp. 322–335. ACM Press, New York (2006)
4. Cristian Cadar, D.E., Dunbar, D.: Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: *OSDI, San Diego, CA, USA (December 2008)*
5. Godefroid, P.: Random testing for security: blackbox vs. whitebox fuzzing. In: *RT 2007: Proceedings of the 2nd international workshop on Random testing*, p. 1. ACM, New York (2007)
6. Godefroid, P., Klarlund, N., Sen, K.: Dart: Directed automated random testing. *ACM SIGPLAN notices* 40(6), 213–223 (2005)
7. Godefroid, P., Levin, M., Molnar, D.: Automated whitebox fuzz testing. In: *Proceedings of the Network and Distributed System Security Symposium (2008)*
8. Jones, S., Arpaci-Dusseau, A., Arpaci-Dusseau, R.: Antfarm: Tracking processes in a virtual machine environment. In: *Proceedings of the USENIX Annual Technical Conference, USENIX 2006 (2006)*
9. Ku, K., Hart, T.E., Chechik, M., Lie, D.: A buffer overflow benchmark for software model checkers. In: *ASE 2007: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pp. 389–392. ACM Press, New York (2007)
10. Lu, S., Zhou, P., Liu, W., Zhou, Y., Torrellas, J.: Pathexpander: Architectural support for increasing the path coverage of dynamic bug detection. In: *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, Washington, DC, USA*, pp. 38–52. IEEE Computer Society Press, Los Alamitos (2006)
11. Molnar, D., Li, X., Wagner, D.: Dynamic test generation to find integer bugs in x86 binary linux programs
12. Nethercote, N., Seward, J.: Valgrind: a framework for heavyweight dynamic binary instrumentation. *SIGPLAN Not.* 42(6), 89–100 (2007)
13. Ramsey, N., Fernández, M.F.: Specifying representations of machine instructions. *ACM Trans. Program. Lang. Syst.* 19(3), 492–524 (1997)
14. Sen, K., Marinov, D., Agha, G.: Cute: a concolic unit testing engine for c. In: *ESEC/FSE-13: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pp. 263–272. ACM Press, New York (2005)

Measuring Information Flow in Reactive Processes

Chunyan Mu

King's College London,
The Strand, London WC2R 2LS
{chunyan.mu}@kcl.ac.uk

Abstract. This paper outlines an approach for measuring information flow within reactive probabilistic systems. First, we present the probabilistic model of reactive labelled transition system with input-output actions. Second, we present the language and semantics for simple reactive processes, and investigate the quantified information flow analysis over this semantics. Third, we define a metric over the semantics and then present a method to compute the leakage in reactive processes. The metric we considered is the square root of the Jensen-Shannon divergence: the quantitative information is contained in the distance between state transformations given by a process metric. Finally, we show that there is a connection between our leakage definition and mutual information in the framework of information theory.

1 Introduction

Information flow measurement has recently become an attractive research topic in the security community. The goal of information flow security in this content is to guarantee that information propagates throughout the execution environment without security violations such that not too much secure information is leaked to public outputs. Traditionally, the approach of information flow security was based on *non-interference* [1], which enforces that there is no secure information about the high inputs can be deduced by observing the low outputs. However, non-interference is too restrictive, and it is too hard to write useful programs in the real world. We therefore consider a new policy to relax non-interference: the program is secure if the amount of information flow from high (confidential) to low (public) is *not too much* from a quantitative point of view. The precursor for this work was that of Denning in the early 1980's. Denning [2] suggested that the data manipulated by a program can be typed with security levels, and first explored the use of information theory as the basis for a quantitative analysis of information flow in programs. However, she did not suggest how to automate the analysis or attempt to make the analysis formal and complete. Millen [3] first built a formal correspondence between non-interference and mutual information, and established a connection between information theory and state-machine models of information flow in computer systems. Wittbold and Johnson [4] gave an analysis of certain combinatorial theories of computer

security from information-theoretic perspective and introduced non-deducibility on strategies due to feedback and internal non-determinism. There has been much recent work in the information theoretic based foundations of quantitative information flow computation [5,6]. Most of the work in this area to date has concentrated on simple programs in simple imperative languages. However, real world programs normally allow input/output, and behave as a reactive system. It is important to consider a quantitative analysis over reactive systems in the computational world. There also have been several attempts on probabilistic and concurrent systems: Di Pierro, Hankin and Wiklicky [7] gave a definition of probabilistic measures on flows in a probabilistic concurrent constraint language where the interference came via probabilistic operators. However, the approach of approximate non-interference presented in this paper is based on the specific probabilistic declarative language PCCP. It seems difficult to automate in other framework. Gavin Lowe [8] measured information flow in CSP by counting refusals. He devised a formal definition of information quantity transmitted from a high level user to a low level user in a computing system. The definition was based on the number of different behaviours of High that can be distinguished from Low's point of view. Like other quantitative definitions, Lowe's definition was based on Shannon's information theory. However, this approach did not consider probabilistic behaviours. Boreale [9] studied the quantitative models of information leakage in the process calculi by applying an information theoretic framework. The *absolute leakage* measured in bits, present the absolute leakage of zero precisely when it satisfies secrecy. The *rate of leakage*, measured in bits per action, presented the maximum information extracted by repeated experiments coincided with the absolute leakage of the process. A weakness of the ratio formulation was that it was difficult to apply to recursive processes.

All to work to date suffers several problems: some of the works provided reasonable analysis on simple program in simple imperative languages, but did not work for programs with complex behaviours like interactions [5,6,10]; some of the works was able to process interactions but the reasonability and completeness of the approach was somewhat weak [8,7]. In this paper, we consider quantitative information flow in reactive systems with input, output, and probabilistic behaviours. The basic concept of our work is that the quantity of information flow is considered by looking at the different behaviours of a high user from a low user's distribution-based observations. We introduce a method to provide a quantitative analysis of information flow for reactive processes due to metric spaces on the process domain. A metric space is built over the execution of the programs via the semantics defined, and the information flow is measured via metrics. The metric we choose here corresponds to the framework of information theory. The attack model in our system considers situations in which a sequence of confidential inputs can be fed into the processes or programs. The attacker can communicate with the program via a set of input-output behaviours. The input-output actions are guarded by probabilistic choices which are following probability distributions. In other words, to capture the secure information flows, we consider the input-output actions with different security levels: high and low

which are governed by high level users and low level users respectively. Low level users are not allowed to observe high level actions but not vice versa. Executing the program produces a set of distribution-based traces in which only low level input-output actions are visible. By observing the visible traces of the program, the attacker tries to collect and deduce some confidential information via the observations. The model we applied for measuring secure information flow within reactive processes is based on probabilistic labelled transition system. The notation of observations is used to provide a basis for recording the history traces of behaviours from the view of low users. Intuitively, the system produces a set of weighted observation trees. Next, inspired by the methodology introduced by [11][2], we define a process domain based on metric spaces, and the metric is with respect to Jensen-Shannon divergence. We use this metric to compute the distance between the different views of the low users due to different behaviours of the high user. We then introduce a method to quantify the secure information flow within processes based on such distances: the quantitative information is contained in the distance within state transformations of the tree set given by a process metric. There are many metrics can be used to measure the distance of distributions, we show that the Jensen-Shannon divergence is a suitable measure of the information flow quantity. To show the intuition behind our method, we discuss that there is a connection between our definition and mutual information in the framework of information theory. We believe our approach provides a reasonable measurement on secure information flow in processes.

The rest of the paper is organized as follows. Section 2 explains the probabilistic model of reactive processes. In Section 3, we present a simple language and semantics for reactive probabilistic processes. Section 4 introduces the method for leakage computation over reactive processes. Finally, we draw conclusions in Section 5.

2 Reactive Probabilistic Labelled Transition System

This section presents a model of reactive probabilistic labelled transition systems. We consider our probabilistic model to be reactive in the sense that the system can react to the environment if fed with a set of high inputs equipped with a probability distribution: by executing a set of low level input-output actions, the system produces a set of observation trees in the way of resulting distributions to the outside.

2.1 Reactive Probabilistic Labelled Transition System

First of all, the model of quantitative reactive systems considered here is based on Probabilistic Labelled Transition Systems (PLTS). In order to consider probabilistic behaviour and information flow measurement, we consider probabilistic labelled transition systems incorporating probability distributions. A *probability distribution* on a set M is a function $f : M \rightarrow [0, 1]$ such that the set $\{m \in M | f(m) > 0\}$ is finite and $\sum_{m \in M} f(m) = 1$. Intuitively, probabilistic

labelled transition systems are labelled transition systems with probabilities attached to each transition, such that transitions are considered as $P \xrightarrow{a}_\mu Q$, denoting P performing an a labelled transition and then behaving as the state Q with probability μ . A formal definition based on Larsen and Skou's [13] probabilistic model is presented as follows.

Definition 1 (Probabilistic Labelled Transition System). *The probabilistic labelled transition system is given as a triple $PLTS = (T, \Sigma, \mu)$, where T is a set of states, Σ is a set of actions, μ is a family of probability distributions, such that $\mu : T \rightarrow \Sigma \rightarrow (T \rightarrow [0, 1])$. Specifically, $\mu_{p,a} : T \rightarrow [0, 1]$, $\mu_p : \Sigma \rightarrow T \rightarrow [0, 1]$, where for any $a \in \Sigma$ and p is a state that can perform the action a , indicating the possible next states and their probabilities after p has performed a , i.e. $\mu_{p,a}(q) = \lambda$ means that the probability that p becomes q after performing a is λ . Furthermore, $\forall p \in T$ and can perform action a , $\sum_{p' \in T} \mu_{a,p}(p') = 1$, i.e., $\mu_{p,a}$ is a probability distribution.*

Second, to allow reactive behaviours, following [12], we consider that the transition relation \rightarrow is between a set of states and *certain sets*. The sets are defined as a set of a pair consisting actions (Σ) and probability distribution on states ($\mu(\mathcal{R})$): $\{\Sigma \times \mu(\mathcal{R})\}$, which must satisfy the *reactiveness condition*: $X \subseteq \Sigma \times \mu(\mathcal{R})$ is said to satisfy the reactive condition if for any $(a_1, s_1), (a_2, s_2) \in X$ either $a_1 \neq a_2$ or $(a_1, s_1) = (a_2, s_2)$. The definition of the reactive probabilistic labelled transition system is presented as follows on the basis of Norman's definition [14].

Definition 2 (Reactive Probabilistic Labelled Transition System). *A simple reactive probabilistic transition system is defined as a tuple $(\mathcal{R}, \Sigma, \rightarrow)$, where \mathcal{R} is a set of states, Σ is a finite set of actions, and \rightarrow is a transition relation*

$$\rightarrow \subseteq \mathcal{R} \times \wp(\Sigma \times \mu(\mathcal{R}))$$

satisfying: for all $E \in \mathcal{R}$ there exists $S \in \wp(\Sigma \times \mu(\mathcal{R}))$ such that $(E, S) \in \rightarrow$, written as $E \rightarrow S$, where $\wp(\cdot \times \cdot)$ denotes the power set of operators restricted to only finite subsets of Cartesian products satisfying the reactiveness condition.

Let us consider an example of the application of the RPLTS. Consider any $S \in \wp(\Sigma \times \mu(\mathcal{R}))$ as a reactive probabilistic process in which the first move of its behaviour (input action) is made by external choice under a distribution. The initial high input actions set $?H = \{h_1, \dots, h_m\}$ feeds into the system. For any $1 \leq i \leq m$ and $F \in \mathcal{R}$, the probability of $S = \{(h_1, w_1), \dots, (h_m, w_m)\}$ performing the action h_i as their initial move and then behaving as F_i is also given by $w_i(F)$ where $w_i(F) \in \mu(\mathcal{R})$. Intuitively, the RPLTS produces a set of trees. Note that $\{w_i | 1 \leq i \leq m\}$ is a probability distribution thus $\sum_{i=1}^m w_i = 1$. The action for a particular channel is incorporated into a distribution function on the events that occur on the channel. The execution of the reactive processes can be viewed as action-guarded and is on the basis of the probability distributions. The resulting distributions are obtained by executing a sequence of input-output actions, and can be viewed as the reaction of the system in the way of sets of probabilistic

synchronisation sub-trees due to each high-input triggered interaction. We thus define distribution-based observations to capture the transformation of each visible interaction step of the processes. At the end of execution a full description of all the sub-trees is obtained in the form of observations based on probability distributions. The notation of observations will be further discussed in Section 2.3.

2.2 The Security Model

The environment is high and low users: low can not observe high inputs and outputs, but high can observe low. In addition, low knows text or description of the program. The way of interaction between users and the system is based on the input-output actions over channels with security levels. We consider two levels: H and L , where H denotes high-level confidentiality and L denotes low-level confidentiality. On the other hand, in order to simply concentrate on the quantitative analysis of secure information flow, we consider a partition over the actions (labels) as: input $?A$, output $!A$ and internal τ . Put two kind of partitions together we have: $\Sigma ::= ?A_H \mid !A_H \mid ?A_L \mid !A_L \mid \tau$. Internal action τ can not be seen from the outside and happens automatically. Low level input and output actions are visible to the external environment.

2.3 Observations

Assume there are a sequence of high inputs to the RPLTS, the low observations are the probability distribution on the low traces due to the high inputs. Information on the projection of the high inputs from the trace can be deduced from these observations. Observations are used to record the history of *observable* transformations of the system on each interaction step during the executions due to the high inputs, *i.e.* a sequence of visible communications that the system might communicate. The observation set is generated by the system, which is defined as a map from a set of states \mathcal{R} to a probability distribution of observable behaviours on the experiments by performing a set of input actions $?H$: $\mathcal{R} \rightarrow (?H.T \rightarrow [0, 1])$. We put $?H.\perp = 1$, where \perp denotes the case of inactive processes. Each set of high inputs $?H$ introduces an interaction step. During this interaction step, low users are communicating with the system via input-output actions. The system therefore produce an observation tree due to such behaviours of the low traces. The next turn of high inputs $?H'$ starts another interaction step and so on. We present the definition of an interaction step and the observation due to each interaction as follows.

Definition 3 (Interaction unit). *We define an interaction unit as the set of the computation steps of processes due to one set of distribution-based high inputs $?H$. The system thus produces a set of visible computational behaviours as a reaction due to such high inputs: $\mathcal{R} \rightarrow (?H.T \rightarrow [0, 1])$, where \mathcal{R} is a set of states, T is the experiment starting with $?H$, and $?H$ is the set of high inputs which starts this interaction and follows a distribution $\{w_i \mid 1 \leq i \leq m\}$, w_i denotes the probability of each input: $0 < w_i \leq 1$, $\sum_{i=1}^m w_i = 1$, and m is the size of the inputs.*

Definition 4 (Observation). *Observation of one interaction unit on a set of states is defined as the set of all E 's finite visible history traces, and is described as a distribution set $\{\pi_i | 1 \leq i \leq n\}$ due to each high input h_i with its weight (probability) w_i at the beginning of this interaction, where π_i is a distribution obtained by the probabilistic computation tree started by $?h_i$. Note that high input h_i also follows a distribution, i.e., $\sum_{i=1}^n w_i = 1$. Let $?H = \{h_i | 1 \leq i \leq n\}$, we have,*

$$\mathcal{O}(?H.E) = \{w_i \cdot \sum_{j=1}^{m_i} p_{ij}.L_{ij}.\perp | 1 \leq i \leq n\}$$

where L_{ij} denotes a set of visible labels (actions) set over the channel L , \perp denotes the action leads to an inactive state. Note that, $\pi_i = \{p_{ij} | 1 \leq i \leq n, 1 \leq j \leq m_i\}$ is a distribution, $\sum_{j=1}^{m_i} p_{ij} = 1$, $\sum_{i=1}^n w_i = 1$, $\sum_{i=1}^n w_i \cdot \sum_{j=1}^{m_i} p_{ij} = 1$.

Let us consider an example to show how the observation works.

Example 1. Consider simple process E with one interaction in Figure 1. $?h_1.E = \frac{1}{3}b.c.\perp + \frac{2}{3}d.e.\perp$, $?h_2.E = \frac{2}{3}b.(\frac{1}{2}d + \frac{1}{2}e).\perp + \frac{1}{3}c.\perp$, and assume $?H = \{h_1, h_2\}$ (where h_1 with weight $\frac{1}{3}$, h_2 with weight $\frac{2}{3}$), and b, c, d, e are visible actions which can be low inputs/outputs.

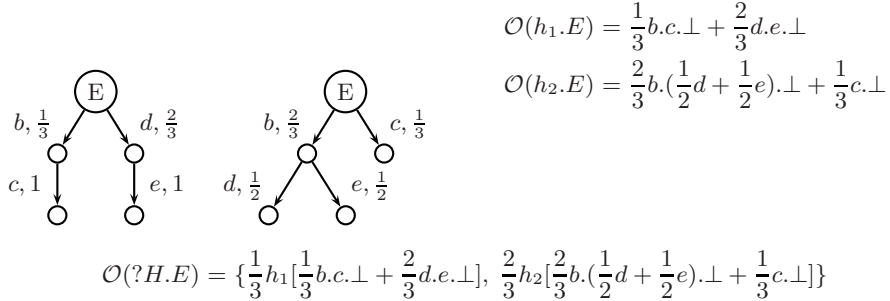


Fig. 1. Example of observations due to interaction unit

3 The Language and Its Semantics of Process Algebra

We consider a CSP-like probabilistic input-output security process algebra, which includes deterministic probabilistic choice, synchronous concurrency, and recursion.

3.1 The Language

The syntax of all expressions is given as follows:

$$F ::= \perp \mid x \mid \sum_{i \in I} a_i.p_i.F_i \mid F_1 \parallel F_2 \mid \mu x.F$$

where \perp denotes the inactive process that does nothing or a state of termination; x denotes a variable, $\sum_{i=1}^n a_i.p_i.F_i$ denotes action-guarded probabilistic choice, where a is a parameterised action set: $a = \bigcup a_i \in \{?H,!H,?L,!L\}$, p_i is the probability of performing a_i and then behaving as F_i , $p_i \in (0, 1]$, $\sum_{i=1}^n p_i = 1$; $F_1 \parallel F_2$ denotes synchronous parallel composition; $\mu x.F$ denotes recursion.

3.2 Operational Semantics

The set of states of the probabilistic labelled transition system $(\mathcal{R}, \Sigma, \rightarrow)$ can be considered as the set of a pair consisting of an action and a probability distribution: $\Sigma \times \mu(\mathcal{R})$, the transition relation is defined as: $\rightarrow \subseteq \mathcal{R} \times (\Sigma \times \mu(\mathcal{R})) \cup \{\emptyset\}$. The pair element of the set is written as: $\pi.a = \sum_{i=1}^n p_i.a_i$, where π denotes the probability distribution, a denotes the parameterised action set: $a \in \{?H,!H,?L,!L\}$, $\sum_{i=1}^n p_i = 1$, $a_i \in a$, p_i denotes the probability of a_i . The action τ is invisible to the external environment and happens automatically. The probabilistic labelled transition system with a set of prefixing inputs can therefore be unwound into a synchronisation tree. The semantics are presented as follows in Table 1, where $E\{F/x\}$ denotes the result of replacing all free occurrences of x in E by F .

Probabilistic choices $a.E$ are guarded by the probabilistic actions following a probability distribution. Consider high inputs as $?H = \{h_i \mid 1 \leq i \leq m\}$, where m is the size of the high inputs, and the probability of performing h_i is w_i . Process E performs h_i and then behaves as E_i . According to the semantics rules, the process is written as $?H.E = \sum_{i=1}^m w_i.h_i.E_i$. The relative probabilistic labelled transition system thus produces a set of probabilistic synchronise trees: $\{w_i.h_i.E_i \mid 1 \leq i \leq m\}$. In the parallel operator, for visible action a , some $\pi_1, \pi_2 \in \mu(\mathcal{R})$: if $E_1 \xrightarrow{a}_{\pi_1} E'_1$ and $E_2 \xrightarrow{a}_{\pi_2} E'_2$. It is clear that $\pi = \pi_1 \pi_2 \in \mu(\mathcal{R})$ is still a distribution. In practice, most recursions are guarded. The first n steps of the behaviour of a guarded recursion $\mu x.F(x)$ can be obtained by unwinding the recursion n times: $F^n(\mu x.F(x))$.

Proposition 1. *For any process, the synchronisation tree produced by its operational semantics forms a RPLTS.*

For proof see our technical report [15].

Table 1. Operational Semantics

Act	$\frac{E \xrightarrow{a_i}_{p_i} E_i}{E \xrightarrow{a}_{\pi} \sum_{i=1}^n p_i.a_i.E_i}$	π denotes the probability distribution: $\{p_i \mid 1 \leq i \leq n\}$ $p_i \in [0, 1]$, $\sum_{i=1}^n p_i = 1$, $a = \{a_i \mid 1 \leq i \leq n\} \in \{?H,!H,?L,!L\}$		
Par	$\frac{E_1 \xrightarrow{\tau} E'_1}{E_1 \parallel E_2 \xrightarrow{\tau} E'_1 \parallel E_2}$	$\frac{E_2 \xrightarrow{\tau} E'_2}{E_1 \parallel E_2 \xrightarrow{\tau} E_1 \parallel E'_2}$	$\frac{E_1 \xrightarrow{a}_{\pi_1} E'_1 \quad E_2 \xrightarrow{a}_{\pi_2} E'_2}{E_1 \parallel E_2 \xrightarrow{a}_{\pi_1, \pi_2} \pi_1 \pi_2.a.(E'_1 \parallel E'_2)}$	$(a \neq \tau)$
Rec	$\frac{}{\mu x.E \xrightarrow{\tau} E[\mu x.E/x]}$			

3.3 Observing Behaviour and Equivalence Relations

A set of observable process traces can therefore be extracted from its operational semantics. For our purpose of quantitative security analysis, a coarser equivalence is considered more satisfactory as it will only distinguish processes that can be distinguished by external low-level observations.

Definition 5 (Probabilistic low bi-simulation). *The low bi-simulation \sim_L is a relation on the set of processes \mathcal{R} : $\{E_i \mid 1 \leq i \leq m\}$ produced by the probabilistic transition system due to a high inputs set $?H = \{w_i.h_i \mid 1 \leq i \leq m\}$, such that, if $E_i \sim_L E_j$ ($1 \leq i, j \leq m$ and $i \neq j$) then:*

$$\forall a \in \{?L, !L\}. \forall S \in L^\sim. E_i \xrightarrow{a}_\mu S \Leftrightarrow E_j \xrightarrow{a}_\mu S$$

where L^\sim denotes the set of low bi-similar classes of \mathcal{R} and $E_i \xrightarrow{a}_\mu S$ if and only if $\mu = \sum \{\mu' \mid E'_i \in S\}$ and $E_i \xrightarrow{a}_{\mu'} E'_i$. Probabilistic processes E_i and E_j are called probabilistic low bi-similar if (E_i, E_j) is contained in some probabilistic low bi-simulations.

4 Information Flow Measurement

We propose to introduce a method for measuring the quantity of information flowed from high-level inputs to public visible observations with respect to the observation tree sets.

4.1 A Metric for Probabilistic Processes

Inspired by the metric space construction for denotational semantics defined by De Bakker & Zucker in [11], and Kwiatkowska & Norman in [12], we introduce a metric with respect to the square root of Jensen-Shannon divergence for the purpose of secure information flow measurement. There are several reasons we choose the JSD as a measure of the difference between distributions. First, JSD is related to other information-theoretical functionals, such as the relative entropy or Kullback Leibler distance. It therefore shares their mathematical properties as well as their intuitive interpretability [16]. Unlike the Kullback Leibler (KL) distance, it is symmetric, always well defined and bounded ($0 \leq \text{JSD} \leq 1$). Second, our system produces a set of transition trees with regard to the weighted high input actions, which may contains more than two elements. JSD can be generalised to measure the distance between more than two distributions, and the compared distributions can be weighted. Third, the square root of JSD ($\sqrt{D_{\text{JS}}}$) is a true metric for the probabilistic distributions space. The $\sqrt{D_{\text{JS}}}$ verifies the triangle inequality, which provides us a potential way to consider the leakage bounds for reactive processes.

Consider m distributions $P^{(1)}, P^{(2)}, \dots, P^{(m)}$ and let $w^{(1)}, w^{(2)}, \dots, w^{(m)}$ denote the corresponding weights. The Jensen-Shannon distance [17] between the m distributions $P^{(1)}, \dots, P^{(m)}$ with weights $w^{(1)}, \dots, w^{(m)}$ is given by

$$D_{\text{JS}}(P^{(1)}, P^{(2)}, \dots, P^{(m)}) = \mathcal{H}\left(\sum_{j=1}^m w^{(j)} P^{(j)}\right) - \sum_{j=1}^m w^{(j)} \mathcal{H}(P^{(j)})$$

To build a true metric space over the denotational semantics of process calculi with respect to the information flow measurement, we consider the *square root* of the Jensen-Shannon divergence as a metric as follows:

Definition 6. For a set of processes $f_1, \dots, f_m \in \mathcal{R}$, $d(f_1, \dots, f_m)$ is defined as the square root of the JSD among m distribution trees $P^{(1)}, \dots, P^{(m)}$ with weights $w^{(1)}, \dots, w^{(m)}$, i.e.

$$d(f_1, \dots, f_m) = \sqrt{\mathcal{H}\left(\sum_{j=1}^m w^{(j)} P^{(j)}\right) - \sum_{j=1}^m w^{(j)} \mathcal{H}(P^{(j)})}$$

Proposition 2. For any processes $f_1, \dots, f_m \in \mathcal{R}$, $d(f_1, \dots, f_m), d(f_1, \dots, f_m) = 0$ iff $P^{(1)} \sim_L \dots \sim_L P^{(m)}$.

For proof see our technical report [15].

4.2 Quantity of the Information Flow

We have already built a metric space for the system, which can be used to measure the distances within processes. In this section, we introduce a definition of information flow quantity for reactive processes over the metric space. The definition of leakage needs to capture how much secure information contained in the high input is released to the public output. First let us concentrate on one interaction step. Consider a set of input actions $?H = \{h_1, h_2, \dots, h_m\}$ with distribution π upon process E : $\pi(E)$ assigns a set of weight/probability (w_1, w_2, \dots, w_m) on a set of process trees. The system start to move with the high input actions and then execute a set of actions based on the structure of the transition system, produce a set of weighted trees denoted by $P^{(1)}, P^{(2)}, \dots, P^{(m)}$, and thus generate an observation set: $\{\mathcal{O}_i(E) | 0 \leq i \leq m\}$ due to the tree set. The observation set maps the process E into a resulting distribution. The information leakage is defined as the square of the distance between the resulting distributions on the tree set: $\mathcal{L} = d^2(P^{(1)}, P^{(2)}, \dots, P^{(m)})$.

Definition 7 (Leakage of one interaction). Assume one high input action set $?H = \{h_1, \dots, h_m\}$ operates on processes \mathcal{R} . The observation tree set produced by the system is denoted by $(P^{(1)}, \dots, P^{(m)})$, where $P^{(i)} = h_i \cdot \mathcal{R}_{1 \leq i \leq m}$ with weight w_i describes the probabilistic transformations of each observed tree. The leakage on one interaction due to such processes is defined as the square of the metric between the observed tree set:

$$\mathcal{L} = d^2(P^{(1)}, \dots, P^{(m)}) = \mathcal{H}\left(\sum_{i=1}^m w^{(i)} P^{(i)}\right) - \sum_{i=1}^m w^{(i)} \mathcal{H}(P^{(i)})$$

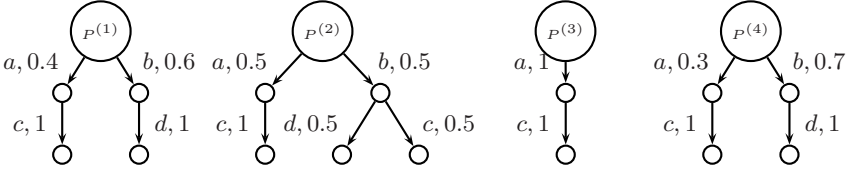


Fig. 2. Example of the leakage computation

Example 2. Assume a high input action set $?H = \{h_i \rightarrow w_i | 1 \leq i \leq 4\}$ as: $h_1 \rightarrow 0.2, h_2 \rightarrow 0.4, h_3 \rightarrow 0.1, h_4 \rightarrow 0.3$. By accepting the set of input actions as the initial move, the system produces a set of process trees as follows:

$$\begin{aligned} \mathcal{O}(P^{(1)}) &= 0.4 \cdot a.c.\perp + 0.6 \cdot b.d.\perp \\ \mathcal{O}(P^{(2)}) &= 0.5 \cdot a.c.\perp + 0.25 \cdot b.d.\perp + 0.25 \cdot b.c.\perp \\ \mathcal{O}(P^{(3)}) &= 1 \cdot a.c.\perp \\ \mathcal{O}(P^{(4)}) &= 0.3 \cdot a.c.\perp + 0.7 \cdot b.d.\perp \end{aligned}$$

We know that the weight of the process trees are: $w^{(1)} = 0.2, w^{(2)} = 0.4, w^{(3)} = 0.1, w^{(4)} = 0.3$. According to our definition of information flow quantity over processes, we have:

$$\mathcal{L} = d^2(P^{(1)}, P^{(2)}, P^{(3)}, P^{(4)}) = \mathcal{H}\left(\sum_{i=1}^4 w^{(i)} P^{(i)}\right) - \sum_{i=1}^4 w^{(i)} \mathcal{H}(P^{(i)}) = 0.31$$

Clearly, zero leakage implies the intuition that for different high inputs, the observer can not tell the difference between them by observing the process trees, and therefore it satisfies non-interference. Bigger leakage implies the observer can tell more difference of the process trees due to the high inputs, and thus more secure information is leaked to the public.

4.3 Measuring Information Flow Over Interaction Steps

One interaction step in the reactive labelled transition systems produces a pair set: $\Sigma \times \mu(\mathcal{R})$. Assume the initial high inputs $?H_0$ start the first interaction step as: $?H_0 = \{(w_{01}, h_{01}), \dots, (w_{0m_0}, h_{0m_0})\}$, where $\sum_{i=1}^{m_0} w_{0i} = 1$. Due to such initial high inputs set, the system produces a set of probabilistic sub-trees. Each computation step of each sub-tree is described as a pair set $\Sigma \times \mu(E_0)$, where E_0 denotes the set of the states taking $?H_0$ as the initial move. During the execution of the program, we may have a sequence of high inputs sets as: $\{?H_0, ?H_1, \dots, ?H_k\}$, where $?H_0 = \{(w_{01}, h_{01}), \dots, (w_{0m_0}, h_{0m_0})\}, \dots, ?H_k = \{(w_{k1}, h_{k1}), \dots, (w_{km_k}, h_{km_k})\}$. For each interaction, we have obtained the metric space $(\xi[i], d_i)$, where $\xi[i]$ denotes the set of probabilistic computation sub-trees due to $?H_i, d_i$ denotes the distance between the computational sub-trees, and $0 \leq i \leq k$. We therefore consider the collection of the metric spaces: $\{(\xi[i], d_i) | 0 \leq i \leq k\}$. Each interaction tree may be incorporated with

a probability of this interaction happens and behaves as an active process. The probability of the process taking $?H_i$ can be considered as the product of the probabilities from the root to the current state which is going to take $?H_i$ as the next move, denoted as q_i , where $0 \leq i \leq k$. Such collection describes the history of the sequence of interaction trees with their probabilities. We then define the maximum information flow leakage due the sequence of high inputs sets as the square of the probabilistic sum of the distance between the sub-trees of each interaction step, *i.e.*, $\mathcal{L} \leq (\sum_{i=1}^k (q_i \cdot d_i))^2$.

Example 3. Consider we have a sequence of high inputs set with two elements: the initial one is $?H_0 = \{\frac{1}{3}h_{01}, \frac{2}{3}h_{02}\}$ which is input at the beginning of the program, another one is $?H_1 = \{\frac{1}{2}h_{11}, \frac{1}{2}h_{12}\}$ which is input into the process during the execution of the program. Assume actions a, b, c are visible. Consider the total distribution tree obtained due to the program is as follows:

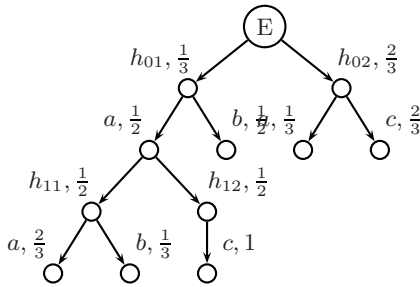


Fig. 3. Example of interactions

$$\begin{aligned} \mathcal{O}(\xi[0]) &= \left\{ \frac{1}{3}h_{01} \cdot \left[\frac{1}{2}a.\perp + \frac{1}{2}b.\perp \right], \frac{2}{3}h_{02} \cdot \left[\frac{1}{3}a.\perp + \frac{2}{3}c.\perp \right] \right\} \\ \mathcal{O}(\xi[1]) &= \frac{1}{6} \cdot \left\{ \frac{1}{2}h_{11} \cdot \left[\frac{2}{3}a.\perp + \frac{1}{3}b.\perp \right], \frac{1}{2}h_{12} \cdot [c.\perp] \right\} \end{aligned}$$

where $\xi[0]$ denotes the truncation of the tree of the process after taking the initial high inputs set but before the second high inputs getting in, $\xi[1]$ denotes the truncation of the tree after taking the second high inputs set with its probability $\frac{1}{6}$ due to current state. Let d_0 denote the distance within $\xi[0]$, and d_1 denote the distance within $\xi[1]$, the maximum information flow quantity from the sequence of high inputs set $?H_0, ?H_1$ to the outside is computed by: $d_0 = \sqrt{0.459} = 0.677$, $d_1 = 1$, $\mathcal{L} \leq (d_0 + \frac{1}{6} \cdot d_1)^2 = 0.712$.

Definition 8 (Leakage upper bound over multi-interaction steps). Assume we have a sequence of high inputs sets, which are fed into the program at the beginning of and during the execution of the program: $\{?H_0, ?H_1, \dots, ?H_k\}$, $?H_0 = \{(w_{01}.h_{01}), \dots, (w_{0m_0}.h_{0m_0})\}$, \dots , $?H_k = \{(w_{k1}.h_{k1}), \dots, (w_{km_k}.h_{km_k})\}$, assume the probabilities of taking $?H_0, \dots, ?H_k$ and behaving as an active process are q_0, q_1, \dots, q_k . The observation tree sets obtained are:

$$\left\{ \{P^{(01)}, \dots, P^{(0m_0)}\}, \{P^{(11)}, \dots, P^{(1m_1)}\}, \dots, \{P^{(k1)}, \dots, P^{(km_k)}\} \right\}$$

We define the information leakage upper bound of this program from the sequence of high inputs sets to the public observer as:

$$\mathcal{L}_{\text{lub}} = \left(\sum_{i=0}^k q_i \cdot d(P^{(i1)}, \dots, P^{(im_i)}) \right)^2$$

4.4 Relationship with Information Theoretic Based Definition

Inspired by the discussion in [16], in this section, we consider a connection between our leakage computation and mutual information to show some intuitions of our method. When only one interaction happens in our reactive probabilistic transition system, the process can be viewed as a batch program produced a set of public outputs in the way of distribution given a set of high inputs under any distribution, *i.e.* a distribution transformer over one interaction from the denotational point of view [10]. Let us concentrate on the case of one interaction step. We have discussed that the observation set provides a set of weighted resulting distributions. Let us denote the observation set due to the truncation of the viewed interaction step as a random variable $\mathcal{O} = \{o_1, o_2, \dots, o_k\}$. Suppose that the sequence of \mathcal{O} is divided into m subsequences: $\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \dots, \mathcal{T}^{(m)}$ with probability $w^{(1)}, w^{(2)}, \dots, w^{(m)}$ based on the distribution of the high input action set which starts this interaction. Let us consider a random vector (o, t) where random variables $o \in \mathcal{O}$ and $t \in \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \dots, \mathcal{T}^{(m)}\}$ are generated as follows: o denotes the observing element locates at t , t denotes the high subsequence that leads to the process tree containing observing point o . Let p_{ij} denotes the joint probability $o = o_i$ and $t = \mathcal{T}^{(j)}$ for $i = 1, 2, \dots, k$ and $j = 1, 2, \dots, m$,

and it can be viewed as: $\begin{pmatrix} p_{11} & p_{12} & \dots & p_{1m} \\ p_{21} & p_{22} & \dots & p_{2m} \\ \dots & \dots & \dots & \dots \\ p_{k1} & p_{k2} & \dots & p_{km} \end{pmatrix}$. Then we get that the random vari-

able o assuming the values o_1, o_2, \dots, o_k with probability p_1, p_2, \dots, p_k . We also get that the random variable t if assuming the values $\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \dots, \mathcal{T}^{(m)}$ with probability $w^{(1)}, w^{(2)}, \dots, w^{(m)}$ where the marginal probability p_i and $w^{(j)}$ are given by $p_i = \sum_{j=1}^m p_{ij}$, $w^{(j)} = \sum_{i=1}^k p_{ij}$, for $i = 1, 2, \dots, k$ and $j = 1, 2, \dots, m$. Intuitively, $p_i = \sum_{j=1}^m w^{(j)} p_i^{(j)}$ defines the probability of finding o_i in the whole tree sequence, *i.e.* $p_i^{(j)} = \sum_{i=1}^k \frac{p_{ij}}{w^{(j)}}$ is the normalised probability of finding o_i in $\mathcal{T}^{(j)}$. Therefore $P^{(j)} = \sum_{i=1}^k p_i^{(j)}$. We consider the Jensen-Shannon distance of $P^{(1)}, P^{(2)}, \dots, P^{(m)}$ due to the truncation of one interaction:

$$D_{\text{JS}}(P^{(1)}, P^{(2)}, \dots, P^{(m)}) = \mathcal{H}\left(\sum_{j=1}^m w^{(j)} P^{(j)}\right) - \sum_{j=1}^m w^{(j)} \mathcal{H}(P^{(j)})$$

$$= \sum_{j=1}^m w^{(j)} \sum_{i=1}^k p_i^{(j)} \log_2 p_i^{(j)} - \sum_{i=1}^k \left(\sum_{j=1}^m p_i \right) \log_2 \left(\sum_{j=1}^m p_i \right)$$

Intuitively, $P^{(j)} = \sum_{i=1}^k p_i^{(j)}$ is the probability of finding o in all subsequence of t . If $P^{(1)} \sim_L P^{(2)} \sim_L \dots \sim_L P^{(m)}$ then it is easy to understand that the identity of o does not tell us anything about the identity of high subsequence t from which observing point o is observed, as the probability distribution of o is identical in all subsequence t , *i.e.* by observing visible actions, we can not get any knowledge about high-level input.

Next, let us consider the mutual information in o about sequence t due to high input, which quantifies the amount of information we obtained from learning the identity of the observing element o about the identity of that subsequence t from which element o was observed. It can be mathematically proven [16] that the mutual information in o about t is identical to the mutual information in t about o , and hence we can state that the Jensen-Shannon divergence D_{JS} quantifies the amount of information we obtain from learning the identity of the chosen element o about the identity of the subsequence t . The mutual information \mathcal{I} in o about t is defined by Shannon [18]:

$$\begin{aligned} \mathcal{I} &= \sum_{i=1}^k \sum_{j=1}^m p_{ij} \log_2 \frac{p_{ij}}{w^{(j)} p_i} = \sum_{i=1}^k \sum_{j=1}^m w^{(j)} p_i^{(j)} \log_2 \frac{p_i^{(j)}}{p_i} \\ &= \sum_{j=1}^m w^{(j)} \sum_{i=1}^k p_i^{(j)} \log_2 p_i^{(j)} - \sum_{i=1}^k p_i \log_2 p_i = D_{JS}(P^{(1)}, P^{(2)}, \dots, P^{(m)}) \end{aligned}$$

Therefore, $D_{JS}(P^{(1)}, P^{(2)}, \dots, P^{(m)})$ over one interaction is equal to the mutual information of o about t , and we obtain Proposition 3.

Proposition 3. *Each interaction truncation of the process can be viewed as a batch program which is given a distribution based high inputs and produces distribution based low observations. For this case, our definition is equivalent to the mutual information between high inputs and low observations.*

Example 4. Let us consider an example with one interaction to see the intuitions. Assume we have two possible high input actions with weights $w^{(1)} = 0.4$, $w^{(2)} = 0.6$ as the initial move of the process. The system thus produces trees $t \in \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}\}$ due to such inputs. Assume that the observation over the first tree $\mathcal{T}^{(1)}$ with weight $w^{(1)} = 0.4$ is as

$$\mathcal{O}(P^{(1)}) = 0.2a.b.\perp + 0.3b.\perp + 0.5a.b.c.\perp, \quad i.e. \quad \begin{matrix} p_1^{(1)} & p_2^{(1)} & p_3^{(1)} & p_4^{(1)} \\ 0.2 & 0.3 & 0.5 & 0 \end{matrix}$$

and the observation over the second tree $\mathcal{T}^{(2)}$ with weight $w^{(2)} = 0.6$ is as

$$\mathcal{O}(P^{(2)}) = 0.3a.b.\perp + 0.4b.\perp + 0.2a.b.c.\perp + 0.1d.\perp, \quad i.e. \quad \begin{matrix} p_1^{(2)} & p_2^{(2)} & p_3^{(2)} & p_4^{(2)} \\ 0.3 & 0.4 & 0.2 & 0.1 \end{matrix}$$

Therefore, the leakage according to our definition can be computed by:

$$D_{\text{JS}}(P^{(1)}, P^{(2)}) = \mathcal{H}\left(\sum_{j=1}^2 w^{(j)} P^{(j)}\right) - \sum_{j=1}^2 w^{(j)} \mathcal{H}(P^{(j)}) = 0.1034$$

On the other hand, since the joint probability p_{ij} of (o, t) obtained by the distribution trees can be considered as: $\begin{pmatrix} 0.08 & 0.12 & 0.20 & 0 \\ 0.18 & 0.24 & 0.12 & 0.06 \end{pmatrix}$, we then can compute the public output p_i as $(0.26 \ 0.36 \ 0.32 \ 0.06)$. Therefore, mutual information in o about t , *i.e.* the mutual information between public output view $Y = \{p_1, p_2, p_3, p_4\}$ and high input with weight $X = \{w^{(1)}, w^{(2)}\}$ is computed by: $\mathcal{I}(X; Y) = \sum_{i=1}^4 \sum_{j=1}^2 p_{ij} \log_2 \frac{p_{ij}}{w^{(j)} p_i} = 0.1034$.

The example illustrates Proposition 3, and thus shows the intuition of the connection between our method of leakage analysis of interactive processes and the information theoretic based definition for batch programs discussed above.

5 Conclusions

We have introduced a method to measure the information flow within reactive system. The probabilistic system we considered is based on probabilistic labelled transition systems. We apply the framework of Kwiatkowska and Norman's metric probabilistic semantics [12], and investigate the quantified security properties over this semantics. We define a metric over the semantics and develop a method to compute the information flow quantity over interaction steps in reactive processes. It is shown that there is a connection between our leakage definition and the framework of information theory and non-interference. We have come out with a novel way of binding the leakage from reactive program system which has RPLTS semantics. This is a big step forward. The outcome is that we get estimating an upper bound on the leakage on reactive processes. However, our observer is very strong. Similar to Boreale's work [9], the observers can observe all the possible actions of the system. Another weakness of our approach is that the leakage is a function of the semantics, in general, it is not executable. A suitable approximation is required. For future work, we plan to look a way to weaken the observers, *e.g.* the observers only can observe low output or some other restrictions. We also want to investigate developing algorithm which is able to compute approximation on the leakage upper bound.

Acknowledgments. I am grateful to David Clark for helpful comments on this work. This work is funded by the EPSRC grant EP/C009967/1 Quantitative Information Flow and Royal Society Project Information Flow in Process Algebras. Also thanks to CREST centre at King's College London for their support.

References

1. Goguen, J., Meseguer, J.: Security policies and security models. In: IEEE Symposium on Security and privacy, pp. 11–20. IEEE Computer Society Press, Los Alamitos (1982)

2. Denning, D.E.R.: *Cryptography and Data Security*. Addison-Wesley, Reading (1982)
3. Millen, J.: Covert channel capacity. In: *Proceeding IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, Los Alamitos (1987)
4. Wittbold, J.T., Johnson, D.M.: Information flow in nondeterministic systems. In: *IEEE Symposium on Security and Privacy*, pp. 144–161 (1990)
5. Clark, D., Hunt, S., Malacaria, P.: Quantitative analysis of the leakage of confidential data. *ENTCS*, vol. 59. Elsevier, Amsterdam (2002)
6. Malacaria, P.: Assessing security threats of looping constructs. In: *POPL*, Nice, France, pp. 225–235. ACM Press, New York (2007)
7. Pierro, A.D., Hankin, C., Wiklicky, H.: Approximate non-interference. In: *CSFW*, pp. 3–17 (2002)
8. Lowe, G.: Quantifying information flow. In: *Proceedings IEEE Computer Security Foundations Workshop*, pp. 18–31 (2002)
9. Boreale, M.: Quantifying information leakage in process calculi. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) *ICALP 2006*. LNCS, vol. 4052, pp. 119–131. Springer, Heidelberg (2006)
10. Mu, C., Clark, D.: Quantitative analysis of secure information flow via probabilistic semantics. In: *ARES*, pp. 49–57. IEEE Computer Society Press, Los Alamitos (2009)
11. de Bakker, J.W., Zucker, J.I.: Processes and the denotational semantics of concurrency. *Information and Control* 54, 70–120 (1982)
12. Kwiatkowska, M., Norman, G.: Probabilistic metric semantics for a simple language with recursion. In: Penczek, W., Szalas, A. (eds.) *MFCs 1996*. LNCS, vol. 1113, pp. 419–430. Springer, Heidelberg (1996)
13. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing (preliminary report). In: *POPL*, pp. 344–352 (1989)
14. Norman, G.: *Metric Semantics for Reactive Probabilistic Processes*. PhD thesis, School of Computer Science, University of Birmingham (1997)
15. Mu, C.: Jensen-shannon divergence as a measure of information flow in reactive processes. Technical Report TR-09-07, Department of Computer Science, King's College London (2009)
16. Grosse, I., Bernaola-Galván, P., Carpena, P., Román-Roldán, R., Oliver, J., Stanley, H.E.: Analysis of symbolic sequences using the jensen-shannon divergence. *Phys. Rev. E* 65, 041905 (2002)
17. Lin, J.: Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory* 37, 145–151 (1991)
18. Shannon, C.E.: A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.* 5, 3–55 (1948)

Trusted Isolation Environment: An Attestation Architecture with Usage Control Model

Anbang Ruan^{1,2}, Qingni Shen^{1,2}, Liang Gu^{2,3}, Li Wang^{1,2},
Lei Shi^{2,3}, Yahui Yang^{1,2}, and Zhong Chen^{1,2,3}

¹ School of Software and Microelectronics, Peking University, Beijing, China

² Key Laboratory of High Confidence Software Technologies,
Peking University, Beijing, China

³ Institute of Software, School of EECS, Peking University, Beijing, China
{ruanab, shenqn, wangl, shil, chen}@infosec.pku.edu.cn,
yhyang@ss.pku.edu.cn, guliang05@sei.pku.edu.cn

Abstract. The Trusted Computing Group (TCG) proposed remote attestation as a solution for establishing trust among distributed applications. However, current TCG attestation architecture requires challengers to attest to every program loaded on the target platform, which will increase the attestation overhead and bring privacy leakage and other security risks. In this paper, we define a conceptual model called the Trusted Isolation Environment (TIE) to facilitate remote attestation. We then present the implementation of TIE with our tailored Usage CONtrol model (UCON_{RA}) and a set of system-defined policies. With its continuous and mutable feature and obligation support, we construct the TIE with flexibility. Lastly, we propose our attestation architecture with UCON_{RA} gaining the benefits of scalable and lightweight.

Keywords: Remote attestation, trusted computing, usage control, MAC model, isolation.

1 Introduction

Problems brought about by malicious applications are becoming severer as the widely use of distributed applications. There are increasing needs to securely identify the software stacks running on remote systems. People may want to determine that services advertised by a remote server really exist and that the system is not compromised, e.g. the E-banking they are interacting with has not been tampered with. Meanwhile, a server needs to confirm that its remote clients will behave in the expected way. For instance, a game client is not fabricated and will abide the game's rules.

To equip applications (or *challengers*) with the capability of identifying the genuine behaviors of their interacting peers (or *targets*), the Trusted Computing Group (TCG) [1] introduced remote attestation as an important feature in TCG specifications to validate the configuration states of remote platforms (or *target platforms*). TCG-compliant system can build a trust chain from an immutable booting base called the Core Trusted Root for Measurement (CRTM), ultimately to applications [2] by iteratively measuring

each part of the boot sequence. With the measurement of the trust chain, corresponding measurement logs and a known-good measurements list, the challenger can then verify whether the target platform has specific configurations.

In order to discover the existence of malicious programs, the TCG attestation requires the challenger to verify every program loaded on the target platform [1], [2], [3], [4], [5], namely the target, the target's dependencies and all other unrelated programs. However, to attest to those unrelated programs will not only introduce unnecessary overheads, but also make the list of known-good measurements unmanageable [3]. Moreover, to expose information of all programs running on the system will incur privacy leakage and increase security risks [4], [5]. A variety of approaches have been proposed for these deficiencies. However, most of them introduce different limitations. For example, some rely on well-configured security policies on the target platforms [3], [16]. They will introduce extra management complexities especially when the number of applications grows. Another line of works demand modifications to software [13], [14]. Hence they may only be applied to particular kinds of applications. Property-based attestation [4], [5] enhances privacy and scalability for the target platform. However, to clearly define and classify appropriate properties for general applications is still an open issue.

In this paper, we propose a Trusted Isolation Environment (TIE) for improving TCG attestation architecture, with our following efforts: a) With our Usage CONTROL model [6] for Remote Attestation ($UCON_{RA}$) and a set of system-defined $UCON_{RA}$ policies, we construct the TIE for a target application in a dynamic and scalable way, without the needs to manually specify the access control policies for managing complicated access relationship. b) Most operations of the attestation are performed in the target platform, gaining more controllable attestation granularity and timing (with the trustworthy guarantee from underlying trusted hardware [9], [10] and related protection mechanisms [12], [14], [18] to protect the local attestation mechanisms). c) A lightweight and easy-to-implement attestation architecture is proposed in this paper. Moreover, with practices [15], [17] in MAC (Mandatory Access Control) models and enforcement mechanisms, our architecture is easy to be integrated into exist systems.

The remainder of the paper is organized as follows: Section 2 presents the definition and requirements for the Trusted Isolation Environment. Section 3 describes the formal specification for $UCON_{RA}$ and corresponding system-defined policies to implement the semantics of the TIE. Our attestation architecture with $UCON_{RA}$ is then illustrated in Section 4. Section 5 presents a case study on Firefox for our architecture. Section 6 discusses some related issues. Section 7 summarizes the related works and section 8 concludes this paper and presents our future works.

2 Trusted Isolation Environment

Commonly, an application needs to interact with other software components on the same system to implement its own functionalities. For instance, during an application's execution, it may need to load extra libraries or to read some data or configuration files. We denote those components, whose integrity will subvert the genuine behavior of a program, as the program's *dependencies* (or the *operating-system-level dependencies* [8]). They represent the program's functionalities. As the loading process

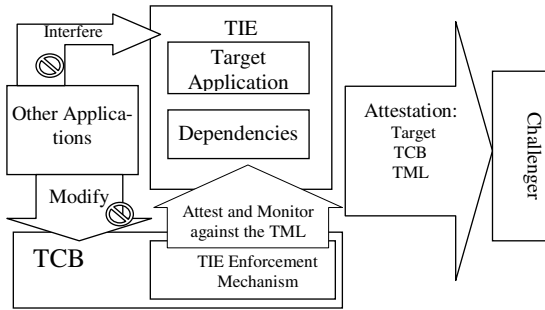


Fig. 1. Remote Attestation with TIE

continues iteratively, a *dependency tree* can be built for a program, with the program as the root of the tree. On the other hand, even if every node on a dependency tree is intact, their behaviors still depend on the genuine execution of their codes or the accurate interpretation of their data. For example, the subversion of the kernel system-call table will tamper with the genuine behaviors of a program even if its codes are not modified. We denote these instruction-level dependencies [8] to implement the behaviors of the target and its dependencies, whether directly or indirectly, as the target's *Trusted Computing Base (TCB)* (see Figure 1), which usually includes the OS kernel, the boot-loader, the BIOS and the CRTM. They represent the capabilities of fulfilling target's functionalities.

In this paper, as depicted in Figure 1, we propose to isolate all entities in a target's dependency tree in a Trusted Isolation Environment (TIE). A TIE for a target can guarantee that, at operating-system-level, a) only entities expected by the target can be imported; b) all entities inside are measured and trusted by the target; c) the integrity of the entities inside is continually protected. A TIE is constructed when a target application is loaded, which is called the TIE Entrance (TIEE). The TIEE can specify an initial Trusted Measurement List (TML), which contains a list of the TIEE's expected dependencies, their genuine measurement values and optional attestation method specifications. Any program loaded in the TIE can also specify its TML to declare its genuine dependencies. Hence, all TMLs define the dependency tree for the target. Entities must be attested to by their parents in the tree when they are entering the TIE (being loaded by entities inside the TIE). All entities can then be trusted by the target in an iterative way and, ultimately, by the challenger. Meanwhile, a TML can also specify rules for inspecting the program's critical state. With the isolation and attestation mechanism of the TIE, a challenger only needs to attest to the target, its TCB and its TML (Figure 1).

2.1 Threat Model

As depicted in Figure 2, adversaries can interfere with the target through three kinds of objects: User Space Objects (USO), Kernel Space Objects (KSO) and Kernel Objects (KO). The USOs are objects existing in the file system (data, libraries). They comprise the most parts of the target's dependency tree. KSOs are kernel services specified to a program, such as IPC objects, semaphore, etc. They also represent the

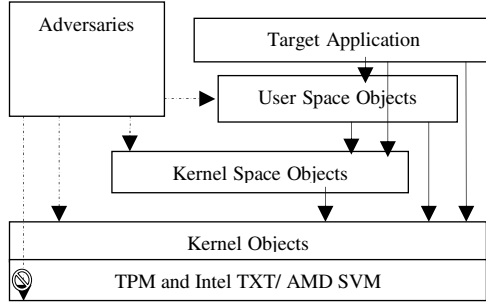


Fig. 2. Threat Model

target's functionalities. Hence they are the target's dependencies. KOs are kernel components (resources and functionalities), e.g. the system call table, kernel codes. They are important parts of the target's TCB. On the other hand, we assume that the adversaries cannot tamper with the secure hardware: the TPM [1], Intel TXT [9] or AMD SVM [10].

Therefore, an isolated and protected environment must be provided for USOs and KSOs. Moreover, the integrity of KOs and the isolation mechanism must also be protected. The former is implemented as the TIE isolation mechanism and the latter as the TIE protection mechanism. Min Xu et al. [12] presented a tailored Usage Control Model (UCON) [6] for kernel-integrity protection ($UCON_{KI}$). They regarded the OS kernel as a set of Kernel Objects (KO) and can then monitor accesses to them with $UCON_{KI}$ rules. Because our solution is also derived from UCON, it is easy to integrate $UCON_{KI}$ into our model. Moreover, the TIE protection mechanism can be facilitated by the trustworthy guarantee from underlying secure hardware [9], [10], and additional protection mechanisms [12], [14], [18]. Therefore, we will focus on the TIE isolation mechanisms in this paper.

3 Usage Control Model for Remote Attestation

In this section, we will first present an overview of the Usage Control model [6] and our Usage Control model for Remote Attestation ($UCON_{RA}$). Afterwards, we will present the formal definition of $UCON_{RA}$, together with our system-defined policies.

3.1 Usage Control Model Overview

Usage control model (UCON) [6] proposed by Jaehong Park and Ravi Sandhu possesses two distinct features over traditional access control models: *continuity* and *mutability*. The *continuity* means that an access decision is not only made before access but also during access and may result in revocation of access permissions if particular conditions are no longer satisfied. The *mutability* means that attributes of subjects or objects may change as side effects of access, which may also result in a change in ongoing or subsequent access decisions. Policy statements in UCON are categorized as *authorizations*, *obligations* and *conditions*. *Authorizations* refer to functional predicates that have

to be evaluated for usage decisions before the access action (*pre-authorization*) or while the action is performing (*on-authorization*). *Obligations* are mandatory actions that a subject has to perform before obtaining (*pre-obligation*) or exercising rights on an object (*on-obligation*).

3.2 TIE Isolation with $UCON_{RA}$

In this paper, we present a tailored Usage CONtrol model (UCON) [6] called $UCON_{RA}$ (Usage CONtrol for Remote Attestation) to construct and maintain a Trusted Isolation Environment (TIE) for the target application and its dependencies (USO and KSO). In our model, a special label called *tag* is defined for TIE identification. We hence specify a simple authorization rule for TIE isolation: subjects can only access the objects with the same tag. We then define a set of policies as the framework for implementing TIE semantics, which are called the *system-defined policies*. We utilize the *mutability* property and *obligation* of UCON for *TIE entrance attestation*: when a subject inside a TIE trying to access entities outside, a *pre-obligation* will first be executed to measure the entities and verify the measurements against the subject's TML. The expected entities will be assigned with the tag of the TIE, allowing all subsequent access requests. Finally, with the *continuity* feature, we define *on-obligation* and *on-authorization* for TIE introspection. They are executed each time when particular rights are exercised, and can move the compromised entities out of the TIE by revoking their tags.

Unlike most MAC models, which manage complicated access relationship, the main concern of $UCON_{RA}$ is to construct and maintain an isolated environment. Therefore, only rules specified above are necessary. Subjects and objects in $UCON_{RA}$ are discovered and labeled at runtime (by obligations), and obligation actions for attestation and monitors are specified in a TML. A TML is defined and issued by the software vendor who has the best knowledge of the software's security requirements and can be refined by the challenger in an attestation session, who can best defined the security requirements for that session. Hence a TIE can be constructed in scalable and flexible way while introducing only a few management overheads.

3.3 $UCON_{RA}$ Definition

In this section, we will examine the components of our tailored usage control model $UCON_{RA}$. We use the UCON definition in [6].

Definition 1. A $UCON_{RA}$ model has seven components:

Subjects (S): active processe;

Objects (O): USO (files), KSO (IPC objects, etc.);

Subject attributes (ATT(S)): tag, text, certificates, TML;

Object attributes (ATT(O)): tag, text, certificates;

Rights (R): access;

Authorizations (A): functional predicates that have to be evaluated for usage decisions.

Obligation (OB): mandatory actions that a subject has to perform before obtaining or exercising rights.

Subjects are mainly active processes. Two kinds of objects are concerned: USO and KSO. Tags specify the TIE the entity (subject or object) belonging to. Text, certificates and the TML are essential for attestation. Only the right “access” is needed for TIE isolation.

An authorization is a functional predicate that has to be evaluated for usage decisions. It can be either evaluated before or while the requested right is exercising, which are called *pre-authorization* and *on-authorization* respectively.

Definition 2. *Pre-authorization:*

$preA(ATT(S), ATT(O), R)$

DEFINITION 3. *On-authorization:*

$onA(ATT(S), ATT(O), R)$

Pre-authorization examines usage requests using $ATT(S)$, $ATT(O)$, and R , and then decides whether the request is allowed or not. On-authorization implements the continuous feature of UCON [6]. It can revoke access right dynamically when specific predicates (onA) are not satisfied. The evaluating process can be performed periodically based on time or event.

Obligations specify mandatory actions (*obligation actions*) that a subject has to perform before obtaining or on exercising rights on an object (*pre-obligation* and *on-obligation* respectively). Both kinds of obligations can be performed with *postUpdate* actions, which update attributes of specific subjects or objects as a side-effect.

Definition 4. *Pre-obligation with postUpdate:*

OBS, OBO, and OB: (obligation subjects, obligation objects, and obligation actions, respectively);

preB : and preOBL, (pre-obligations predicates and pre-obligation actions, respectively);

$preOBL: OBS \times OBO \times OB;$

$preFulfilled: OBS \times OBO \times OB \rightarrow \{true, false\};$

getPreOBL: $S \times O \times R \rightarrow 2^{preOBL}$, a function to select pre-obligations for a requested usage;

$preB(s, o, r) = \bigwedge (obs_i, obo_i, ob_i) \square getPreOBL(s, o, r) preFulfilled(obs_i, obo_i, ob_i);$

$preB(s, o, r) = true$ if $getPreOBL(s, o, r) = \varphi;$

postUpdate(ATT(s)), postUpdate(ATT(o)): an optional procedure to change certain attributes as a consequence of pre-obligations.

The $preB$ predicate evaluates if all the required pre-obligation actions ($preOBL$) are fulfilled by using $preFulfilled$ and returns either true or false. $GetPreOBL$ decides on what kind of actions are required for requests. OBS is the entity who has to perform the action. OBO is the entity on which the action has to be performed. OB represents what has to be performed. The $postUpdate$ specifies the attributes of subjects or objects to update after the obligation.

Definition 5. *On-obligation:*

T, a set of time or event elements;

onB and onOBL, (ongoing-obligations predicates and ongoing-obligation elements, respectively);

$onOBL\ OBS \times OBO \times OB \times T$;
 $getOnOBL: S \times O \times R \rightarrow 2^{onOBL}$, a function to select ongoing-obligations for a requested usage;
 $onFulfilled: OBS \times OBO \times OB \times T \rightarrow \{true, false\}$;
 $onB(s, o, r) = \bigwedge (obs_i, obo_i, ob_i, t_i) \square getOnOBL(s, o, r) onFulfilled(obs_i, obo_i, ob_i, t_i)$;
 $onB(s, o, r) = true$ if $getOnOBL(s, o, r) = \varphi$;

On-obligations fulfill obligation actions, either periodically or continuously, while the rights are being exercised. Most actions are similar with the ones defined in pre-obligation. Particularly, a time parameter T is introduced as part of obligation actions on OBL. T is likely to define certain time intervals that are either time-based or event-based. OnB must be assumed to be true all the time though actual obligation verification intervals can vary.

3.4 System-Defined Policies

To implement the TIE functionalities, we must first confine communications within entities in the same TIE, and then make sure only trusted entity can enter the TIE. Lastly, the dynamic state of a TIE can be introspected. We hence define three policies: pre-authorization for TIE confinement, pre-obligation for TIE entrance attestation, and on-obligation/on-authorization for TIE monitors.

Two kinds of decisions can be defined in $UCon_{RA}$.

Definition 6. *Allowed decision:*

$allowed(s, o, r) \Leftrightarrow predicates$

Definition 7. *Stopped decision:*

$stopped(s, o, r) \Leftrightarrow \neg predicates$

The $allowed(s, o, r)$ indicates that subject s is allowed right r to object o . According to [6], the ‘implies’ connectives is used, which represents the ‘necessary condition’. Henceforth, the decision process can include other rules that might be necessary for finer and richer controls. The ‘stopped(s, o, r)’ indicates rights r of subject s to object o is revoked, and the right-hand-side is a ‘sufficient condition’, by which means the dissatisfied of any predicates may cause the ‘stopped’ decision.

The TIE confinement policy is quite simple: subjects can only access objects with the same tag.

Definition 8. *The TIE confinement policy:*

T is the tag of a subject (s) or an object (o), representing which TIE it is currently belonging to (NULL for none).

Tag: $S \rightarrow T, O \rightarrow T$

$ATT(S) = (Tag)$

$ATT(O) = (Tag)$

$preA1: Tag(S) == Tag(O)$

$preA2: Tag(S) == NULL \ \&\& \ Tag(O) == NULL$

$allowed(S, O, access) \Leftrightarrow preA1 \ || \ preA2$

When a subject in the TIE requests to access an object outside, the TIE isolation mechanism will first decide whether the object is trustworthy by measuring it and

then verifying the measurement against the object's certificate and the certificate against its loader's TML. Moreover, the loader can choose to perform the attestation by itself with only a few changes to its TML as described in Section 5. Expected object will be assigned with the same tag as the subject, by which means to be moved into the TIE. This semantic is implemented by a pre-obligation with post tag update.

Definition 9. *TIE entrance attestation obligation:*

TXT is the text of an object.

L is the Trusted Measurement List of a subject.

C is the certificate of an object.

Text: $O \rightarrow TXT$; TML: $S \rightarrow L$; Certificates: $O \rightarrow C$;

Attest: $((Text(O), Certificates(O), TML(S)) \rightarrow \{true, false\}$

ATT(O) = (Text, Certificates)

ATT(S) = (TML)

OBS = Measurement Agent;

OBO = Text(O);

OB = {Attest};

getPreOBL(s,o,r) = {(measurement agent, text(o), attest)} if (tag(s) != NULL && tag(o) == NULL)

allowed \Leftrightarrow preFulfilled(getPreOBL(s,o,r))

postUpdate(Tag(OBO)): tag(o) = attest(obo, tml(s), certificates(obo)) ? tag(s):NULL;

Monitor obligations can inspect the state of a TIE at runtime. It performs particular actions (specified in TML) while subjects are accessing objects, and updates the attributes when necessary. Meanwhile, monitor authorization can perceive the change of the system state by examine the entities' tags while specified rights are exercising.

Definition 10. *Monitor obligation:*

OBS = Monitor Agent

OBO = {S, O}

OB = {Monitor}

Monitor: predicates must be satisfied.

T = {always}

getOnOBL(s, o, r) = {(monitor agent, obo, monitor, always)}

postUpdate(Tag(OBO)): Tag(OBO) = monitor ? Tag(OBO) : NULL

DEFINITION 11. *Monitor authorization:*

allowed(s, o, r) \Leftrightarrow true

stopped(s, o, r) \Leftrightarrow (Tag(S) != Tag(O)) || (Tag(S) == NULL && Tag(O) != NULL) || (Tag(S) != NULL && Tag(O) == NULL)

According to UCON, on-obligation can also specify allowed and stopped decisions, to make or revoked usage decision directly. However, in our model, we clearly separated the duty of obligations and authorizations. The former are responsible for TIE entrance or exit, and the latter are for isolation. Authorizations make decisions simply by the tag of both subjects and objects. Meanwhile, obligations utilize other attributes, and affect the access decision through altering the tags. The separation of duties makes it easy to simplify our model and system-defined polices, and hence reduces the management complexities.

Obligations above only provide a framework for specifying the occasions to perform particular obligation actions. Obligation actions for particular applications are commonly specified within the TML by their vendors, and can be revised by the challenger. Hence different kinds of monitors and attestation methods can be customized as needed. With the attestation action specifiable in a TML, we can choose the best-fit attestation methods. Moreover, we can specify particular parts of an entity to be attested, achieving the best attestation granularity. Section 5 will present examples for specifying the attestation methods in a TML.

4 Attestation Architecture with UCON_{RA}

In this section, we will first describe our attestation architecture and its functional components. Then we will illustrate how they cooperate in the scenarios of TIE construction and remote attestation.

4.1 Attestation Architecture with TIE

Figure 3 illustrates the overview of our attestation architecture with UCON_{RA}. Just as a typical Mandatory Access Control (MAC) enforcement architecture [17], our architecture includes three main components: Policy Enforcement Point (PEP), Attribute Repository (AR) and Policy Decision Point (PDP). PEP is a module to intercept access request from the processes, to pull subject and object attributes from AR, and to enforce access executions. AR is used to store subject and object attributes. PDP delivers authorization decisions according to policies stored in the Policy DB. Obligations are stored in the Policy DB as well. To improve performance, the Access Vector Cache (AVC) component caches access decisions. The TIE Registry maintains an entry for each TIE, recording information for their lifetimes. Our architecture also contains two components for obligation action execution, namely Measurement Agent and Monitor. The component of Monitor is actually a monitor enforcement driver to execute specific scripts or programs specified by the software vendor or the challenger for monitoring various states of a TIE. The integrity of these monitor actions must also be guaranteed. Lastly, Attestation aGent (AG) collects related information and manages attestation sessions with the remote parties.

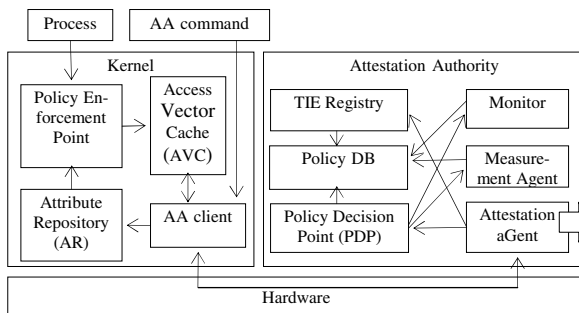


Fig. 3. Attestation Architecture with TIE

Because the attestations for the target's dependencies are performed in the target platform, we move the components for TIE and policy management, obligation action enforcement, and policy decision out of the kernel space into a tiny hypervisor [11], which is called the Attestation Authority (AA), to minimize their TCB and to achieve strong isolation. Therefore an additional component for AA communication is added, named AA Client. Lastly, the AA Command component serves as the interface to user.

4.2 TIE Construction

A TIE is represented as a $UCON_{RA}$ tag. Hence we can control the affiliation of an entity by managing its tag. After we registered the program in the TIE Registry as the TIE Entrance (TIEE), we can initiate it through the AA Command, which will then inform Attestation Authority (AA) to load and parse its Trusted Measurement List (TML) and store obligation actions into Policy DB. AA will measure and load the TIEE and assign it a tag representing the newly created TIE. Afterwards, any entity created by the subjects in the TIE will be assigned with the same tag.

When Policy Decision Point (PDP) receives an access request from a subject inside a TIE to an object outside (with a different or no tag), the attestation obligation is first executed. PDP first searches for particular obligation action, and transfers it (or a default one) to the Measurement Agent (MA), which will attest to the object in the way specified by the obligation action in subject's TML. MA then updates the object's tag in the Attribute Repository (AR) to allow or deny the request. On the other hand, when a subject outside the TIE (untagged) requests to access an object inside (tagged), Policy Enforcement Point (PEP) will simply deny it, or for shared USOs (Section 5), it will copy the object to a location specific to the TIE and redirect the access requests from the tagged subjects to the new copy. The origin one is then unlabeled, enabling the access request from other untagged subjects.

4.3 Remote Attestation

Figure 4 illustrates our remote attestation procedure. Administrator first informs Attestation aGent (AG) to initiate the TIE Entrance (TIEE) and construct the TIE (step 1.1-1.2). When the TIEE connects to a challenger, AG will first transfer the measurement of the TCB (from the CRTM up to the OS kernel and AA) and the target

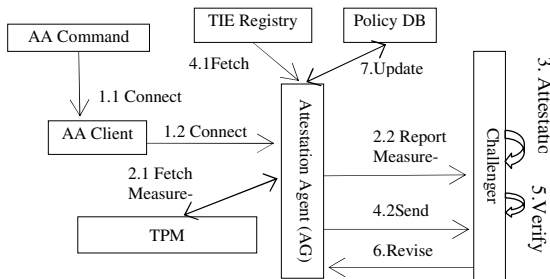


Fig. 4. Remote Attestation Procedure

application to the challenger (step 2.1-2.2) for attestation (step 3), and negotiates the Trusted Measurement List (TML) with the challenger (step 4.1-4.2). The challenger then verifies the TML (step 5) and returns a revised TML to AG when needed (step 6). Afterwards, AG updates the Policy DB in accordance with the new TML (step 7) and then loads the target application. In addition, the PDP and PEP, though not involved in the attestation process directly, are responsible for maintaining the TIE.

5 Case Study

In this section, we will present a case study on Firefox. We will illustrate a TML for the Firefox, and describe its TIE's building and attestation process in our architecture.

5.1 TML Specification for Firefox

Typically, the TML will be defined and signed by the Firefox's vendor. In our case, we downloaded the package of Firefox-3.0.5 from its official site and manually specified its TML by identifying all legal dependencies, calculating their measurements and specifying related measurement methods (if needed). Table 1 shows an abridged view of the TML.

Only the security-related files need to be attested, such as the firefox-bin (the TIEE), the shared libraries (.so files), and all files in the directories of components, modules, and searchplugins. Resource files such as icon files (.gif files) will not affect the genuine behavior of Firefox. Henceforth, their attestation methods and measurements in the TML are specified as NONE. By specifying the Attestation Method entry, our architecture can easily support attestation for particular parts of a file, e.g. for configuration files as browserconfig, we specify that the browser.startup.home page entry must be assigned to the default value.

Web services may demand to load particular plugins, which can also be specified through the TML negotiation phase. For files that might be altered, we specify Mutable flags in their TML entries. When the TIE is exiting, those files are hashed and the measurement entries are updated. Every time files with Shared flags are referenced by entities with a different or no tag, Attestation Agent (AA) will first copy them to a dedicated directory for the current TIE, redirect all existing access requests and untag

Table 1. Abridged TML for Firefox-3.0.5

File Name	Example Measurement	Attestation Method
firefox-bin	C9D250E82A4C3CFED2246E1CF2 A39526E7CC3CC3	Full file
libfreebl3.so	1A34E6AFE7B07FBD1CD3C1D94C 485A68D598B9DC	Full file. Shared
*.gif	NONE	NONE
browserconfig.properties	http://en- US.start2.mozilla.com/firefox?client= firefox-a&rls=org. mozilla-en- US:official	ENTRY: browser.startup.homepage
blocklist.xml	E857979550DF3C2B4C4F474148E BAF98D214C44	Full file. Mutable.

the original one to allow access from those entities (libfreebl3.so). Monitors for state introspecting are usually challenger-specific. Hence they are often specified as obligation actions by the challenger in the TML negotiation phrase. Obligation actions are executed by monitor module specific to a program, which should also be provided by software vendors. They will be invoked by the Monitor component of AA according to the system-defined monitor obligation rules and obligation actions in the TML.

One difficulty is that the Firefox's GUI system is depending on the integrity of xorg server, which is commonly started before the TIE of Firefox. Because our architecture currently only supports the attestation for the files to load, we confine the xorg server in another TIE and add an entry in Firefox's TML to refer to the TML of the xorg server. When Firefox is started, AA will compare the xorg TML reference to the TML of the currently running xorg server. If they match, an authorization policy will be added to the Policy DB, permitting communication between the two TIEs.

5.2 TIE Isolation for Firefox

Firefox-bin is first registered as a TIEE in the TIE Registry together with its TML and then initiated through the AA command, which will inform AA to construct a corresponding TIE. AA first loads the TML, calculates the hash value of firefox-bin and then compares the hash with related entry in the TML. If the values match, firefox-bin is loaded and assigned with a $UCON_{RA}$ tag representing its embracing TIE. Otherwise firefox-bin will be loaded as a normal program. Every time when firefox-bin is trying to access entities with no tag assigned, a measurement obligation of $UCON_{RA}$ is executed by the Measurement Agent to attest to the entities against TML and to assign those expected ones with the same tag. For those illegal ones, a different tag is assigned, avoiding repeatedly attestation.

The TIE can resist following kinds of threats in our case: a) replacing the shared libraries. Any access request to those libraries from programs outside the TIE is denied by the TIE isolation policy (Definition 8). The tag value is calculated with a nonce every time when a new TIE is constructed, hence cannot be predicted by malicious programs. For files can be shared among the system, each TIE keeps a dedicated copy. Hence changes to the origin ones cannot tamper with the TIE. b) Instructing Firefox to load unexpected plugins. Before firefox-bin loads the new plugin (untagged), the TIE entrance attestation obligation (Definition 9) is executed, which hashes the plugin and searches the hash value in TML. When there is nothing found, the request is denied, and the access decision is cached. c) Instructing Firefox to connect to a new URL. Since the specific state of firefox-bin can be monitored (by Definition 10 and 11 respectively), any connection to URLs other than those specified in the TML are denied. More sophisticated monitor rules and monitor enforcement components can be designed by software issuer or negotiated with the challenger.

6 Discussion

Because the semantics of a TIE are implemented as an MAC model, the TIE isolation mechanism is easy to be integrated into exist systems. Only a few modules are needed to be added to the existing systems, with most derived from SELinux [15]. The Attestation Authority (AA) can be hosted in a tiny-hypervisor as Secvisor [11] to

achieve high security guarantee, which introduces no more than 2,000 lines of codes and can be efficient. The main overheads for TIE are the MAC enforcement and decision. They are well studied and can be under good control. Moreover, because only target's dependencies are measured, both measurement and verification efforts are reduced. Although the target platform has to perform extra verifications, they are related lightweight, compared to those saved measurement efforts.

Our scheme does not eliminate software vulnerabilities. Instead, we minimized the ways for malicious entities to utilize vulnerabilities of programs in the trust chain by isolating them in a Trusted Isolation Environment (TIE). Our architecture can guarantee the challenger that the target application will behave as the way it is designed, and the challenger can then decide its trustworthiness according to extra information, e.g. reported vulnerabilities, or specify monitors for filtering and introspecting.

UCON_{RA} only concerns about constructing and maintaining the TIE. Pre-authorizations control communications between entities inside the TIE and entities outside. Entrance obligations deal with importing entities into the TIE. Monitor obligations utilize the feature of continuously for dynamic introspecting. However, since our model is derived from usage control model, which can implement the semantics of most prevalent access control models (e.g. Role Based Access Control) by specifying particular authorizations [6], access control inside a TIE can easily be supported.

The Trusted Measurement List (TML) is specified by the issuer of the target application and can be customized by the challenger through the TML negotiation phase (step 6 in Section 4.3). Hence the dependencies can be clearly defined. Meanwhile, they can also be identified with both static program analysis and runtime monitoring [8]. We also considered many ways to reduce the length of a TML, e.g. TML can be referenced and entities can be copied and dedicated to a particular TIE as the `xorg` and `libfreebl3.so` case in previous section respectively. For complicated applications with various dependencies, the TML may become very large or may refer to many other TMLs on the target platform. However, the complexity is still relative low comparing to the known-good measurement list for these kinds of applications. We will investigate further on reducing the size of TML.

7 Related Work

Terra [7] uses a Trusted Virtual Machine Monitor (TVMM) that provides the semantics of either an “open box” or a “closed box”. The hardware and TVMM can act as a trusted party to allow closed-box to cryptographically identify the software they run to remote parties. However, administrators have to pay extra management overheads to deploy the closed box for each target application. In our scheme, a TIE is constructed at runtime, and administrators only need to register the TIE Entrance (TIEE) and initiate the TIEE. The TIE semantics are implemented as an MAC model; hence they can easily be tailored or enhanced and can be integrated into existing systems with only few efforts, gaining scalable and lightweight. As a result, we avoid the overheads brought by extra virtualization layer, e.g. the hypervisor.

Policy-Reduced Integrity Measurement Architecture (PRIMA) [3] enables the challengers to only verify the applications which are permitted by the policies to interact with the target. Model-based behavior attestation (MBA) [16] attests to the behavior of a model, which is associated with different components of a policy model

and can be attested to separately at runtime. However, as the dimension of the entities scales, the management complexities for MAC policies introduced by them escalate rapidly. The main concern of $UCON_{RA}$ is to construct and maintain an isolated environment. Therefore, only a few policies are needed (Section 3). Tags are specified as a side-effect of measurement, and the measurement is taken automatically as an obligation when an unauthorized access request occurs. Moreover, obligation actions for attestation and monitors are specified in a TML, which are customizable and can be negotiated with the remote party.

BIND (Binding Instructions and Data) [13] attests to only the concerned pieces of codes, which greatly simplifies the verification, instead of the entire memory content. Flicker [14] isolates sensitive code execution with hardware support for late launch and attestation introduced in commodity processors from AMD [10] and Intel [9]. However, BIND and Flicker need developers' supports: only the specially designed application can be supported. In our architecture, all legacy applications are supported.

Property-based attestation (PBA) [4], [5] proposes to attest to specific property of a system without receiving detailed configuration data. However, to identify appropriate properties for every application is a complicated job, which makes the PBA hard to implement. Min Xu et al. [12] presented a tailored $UCON$ model for kernel-integrity protection ($UCON_{KI}$). They regarded OS kernel as a set of Kernel Objects (KO) and protected them by $UCON_{KI}$ rules. While $UCON_{KI}$ mainly considers the kernel integrity protection, our focus is to provide a runtime constructed trusted and isolated environment to facilitate remote attestation. Moreover, $UCON_{KI}$ can be easily integrated into our model to provide a bottom-up protection.

8 Conclusion and Future Work

In this paper, we proposed a conceptual model called the Trusted Isolation Environment (TIE) to facilitate remote attestation. With the TIE, only genuine entities expected by the target application can interact with it. Therefore, challengers only need to attest to the TCB of the target platform, the target, and the target's Trusted Measurement List (TML). We then presented our tailored $UCON_{RA}$ model and properly designed policies. With the continuous and mutable feature and obligation support from $UCON_{RA}$, our architecture can be scalable and lightweight. For future works, we will investigate additional mechanisms to reduce the complexity of the TML.

Acknowledgement. This paper is supported by National Natural Science Foundation of China under Grant No. 60873238.

References

1. Trusted Computing Group (TCG), <https://www.trustedcomputinggroup.org/>
2. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and Implementation of a TCG-based Integrity Measurement Architecture. In: SSYM 2004: Proceedings of the 13th conference on USENIX Security Symposium, Berkeley, CA, USA, p. 16. USENIX Association (2004)

3. Jaeger, T., Sailer, R., Shankar, U.: PRIMA: Policy-Reduced Integrity Measurement Architecture. In: SACMAT 2006: Proceedings of the eleventh ACM symposium on Access control models and technologies, pp. 19–28. ACM Press, New York (2006)
4. Poritz, J., Schunter, M., Van Herreweghen, E., Waidner, M.: Property attestation—scalable and privacy-friendly security assessment of peer computers. Technical Report RZ 3548, IBM Research (May 2004)
5. Sadeghi, A.R., Stübke, C.: Property-based attestation for computing platforms: Caring about properties, not mechanisms. In: The 2004 New Security Paradigms Workshop, Virginia Beach, VA, USA, ACM SIGSAC, September 2004. ACM SIGSAC, ACM Press (2004)
6. Park, J., Sandhu, R.: The UCON_{abc} usage control model. ACM Transactions on Information and Systems Security 7(1) (February 2004)
7. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), pp. 193–206 (2003)
8. Gu, L., Ding, X., Deng, R.H., Xie, B., Mei, H.: Remote Attestation on Program Execution. In: Proceedings of STC 2008, Virginia, USA (October 2008)
9. Intel Corporation: LaGrande technology preliminary architecture specification. Intel Publication no. D52212, May 2006.
10. Advanced Micro Devices. AMD64 virtualization, C.: Secure virtual machine architecture reference manual. AMD Publication no. 33047 rev. 3.01 (May 2005)
11. Seshadri, A., Luk, M., Qu, N., Perrig, A.: SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In: SOSP 2007 Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, Washington, USA (October 2007)
12. Xu, M., Jiang, X., Sandhu, R., Zhang, X.: Towards a VMM-based Usage Control Framework for OS Kernel Integrity Protection. In: Proceedings of SACMAT 2007, Sophia Antipolis, France (2007)
13. Shi, E., Perrig, A., Van Doorn, A.: BIND: A Fine-Grained Attestation Service for Secure Distributed Systems. In: SP 2005: Proceedings of the 2005 IEEE Symposium on Security and Privacy, pp. 154–168 (2005)
14. McCuney, J.M., Parnoy, B., Perrig, A., Reiteryz, M.K., Isozaki, H.: Flicker: An Execution Infrastructure for TCB Minimization. In: Proceedings of EuroSys 2008, Glasgow, Scotland, UK (April 2008).
15. Security-Enhanced Linux (SELinux), <http://www.nsa.gov/selinux/>
16. Alam, M., Zhang, X., Nauman, M., Ali, T., Seifert, J.P.: Model-based Behavioral Attestation. In: SACMAT 2008: Proceedings of the thirteenth ACM symposium on Access control models and technologies, ACM Press, New York (2008)
17. Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., Lepreau, J.: The Flask Security Architecture: System Support for Diverse Security Policies. In: Proceedings of the Eighth USENIX Security Symposium, Aug. 1999, pp. 123–139 (1999)
18. Loscocco, P.A., Wilson, P.W., Aaron Pendergrass, J., McDonnell, D.: Linux kernel integrity measurement using contextual inspection. In: Proceedings of the 2007 ACM workshop on Scalable trusted computing, November 02-02, 2007, Alexandria, Virginia, USA (2007)

Denial-of-Service Attacks on Host-Based Generic Unpackers*

Limin Liu¹, Jiang Ming², Zhi Wang^{2,3}, Debin Gao², and Chunfu Jia³

¹ State Key Lab of Information Security, Graduate University of CAS, China
lmliu@is.ac.cn

² School of Information Systems, Singapore Management University, Singapore
{jiangming,zwang,dbgao}@smu.edu.sg

³ College of Information Technology and Science, Nankai University, China
cfjia@nankai.edu.cn

Abstract. With the advance of packing techniques, a few generic and automatic unpackers have been proposed. These unpackers are designed to automatically unpack packed binaries without specific knowledge of the packing techniques used. In this paper, we present an automatic packer with which packed malware forges spurious unpacking behaviors that lead to a denial-of-service attack on host-based generic unpackers. We present the design, implementation, and evaluation of the proposed packer and malware produced using the proposed packer, and show the success of denial-of-service attacks on host-based generic unpackers.

Keywords: generic unpacker, denial-of-service attack, spurious unpacking behavior.

1 Introduction

Most malware is packed in order to evade malicious content detection. It has been reported that among 20,000 malware samples collected in April 2008, more than 80% were packed by packers from 150 different families [10]. This is further complicated by the ease of obtaining and modifying the source code of various packers. Currently, new packers are created from existing ones at a rate of 10 to 15 per month [17].

A few generic and automatic unpacking techniques have been proposed to unpack packed binaries without specific knowledge of the packing technique used, e.g., OmniUnpack [9], Justin [5], Renovo [8], PolyUnpack [14] and others [1].

* This research was mostly done when the first three authors, Limin Liu, Jiang Ming, and Zhi Wang, were researchers working in Singapore Management University. It was partially supported by National Science Foundation (NSF) China under the agreements 90718005, 70890084/G021102, and 60573015.

¹ For example, OllyBonE from National Center for Biotechnology Information, <http://www.ncbi.nlm.nih.gov>; Deroko, <http://deroko.phearless.org/rce.html>; RL!Depacker, <http://ap0x.jezgra.net/>; QuickUnpack, <http://qunpack.ahteam.org/>; and Universal PE Unpacker, http://www.hex-rays.com/idapro/unpack_pe/unpacking.pdf.

A difficulty that most of these generic unpackers face is to determine the end of the unpacking routine in the execution of the packed malware, which has been proved to be an undecidable problem [14]. Intuitively, the malware might be packed with multiple layers of packers, which makes it difficult to differentiate the end of the unpacking and the start of the execution of the original malware. Many generic unpackers adopt various heuristics to handle this difficulty. However, they may be wrong in determining the end of unpacking [9].

Using the heuristics does not necessarily expose the unpacker and its operating machine to attacks like denial-of-service attacks we propose in this paper, especially if the unpacker runs on a controlled and protected environment, e.g., emulators or virtual machines. However, when the unpacker is running on a real host machine that has to be protected by, e.g., anti-virus software, a denial-of-service attack becomes possible.

Some of the latest and most sophisticated unpackers operate on real host machines instead of emulators or virtual machines to improve efficiency [5,9]. Since they do not run under a controlled and protected environment, anti-virus software is invoked at *all* potential end-of-unpacking points to protect the host machine and to detect the unpacked malware. For example, OmniUnpack [9] invokes anti-virus software to scan the malware whenever a page is written and then executed, and the execution involves a dangerous system call made.

Our proposed denial-of-service attack exploits the requirement of host-based generic unpackers to invoke anti-virus software at all potential end-of-unpacking points during the execution of the packed malware, and the fact that the anti-virus scanning time dominates the overhead of the unpacker (see Section 2). The idea is for the DoS-packed malware to forge a large number of spurious unpacking routines that lead to a large number of invocations of the anti-virus software, which costs lost of resources on the unpacker including CPU cycles, memory, and time.

The automatic packer we propose in this paper attaches the DoS attack code into the executable of malware. The resulting packed malware has the following properties.

- benign: the DoS attack code we add to the malware (not including the original malware) does no harm to the operating system, which makes the packed malware and its partially packed version evade the anti-virus scanning.
- environment-aware: the DoS attack only targets the unpacker and is not executed in normal machines (victims of the original malware).
- semantic-preserving: our packer does not change the semantics of the original malware.
- light-weighted: the overhead when executing on normal machines is small.

In this paper, we present a security vulnerability of host-based generic unpackers, and propose an automatic packer with which packed malware forges spurious unpacking behaviors that lead to a denial-of-service attack on host-based generic unpackers. We implement this packer and try it out on a number of malware. Experiments show that the packed malware successfully differentiates the unpacker and normal environments, and launches DoS attacks on the target unpacker.

In the rest of this paper, we first describe the two types of unpackers available and an analysis of the host-based unpackers (see Section 2). In Section 3, we detail the design of our denial-of-service attack and the implementation of our automatic packer that realizes the attack. Section 4 presents the evaluation of the proposed attack and its implementation.

2 Generic Unpackers and Their Heuristics

When a packed malware is executed, it first dynamically unpacks its malware payload and then executes the malware. Generic unpackers try to extract the malware payload after detecting the end of unpacking. However, determining the end of unpacking is proved to be an undecidable problem [14]. Therefore, generic unpackers either 1) extract the malware payload after it is executed, or 2) have to introduce various heuristics to approximate the end of unpacking. The approach adopted by an unpacker depends on its executing environment. Some unpackers execute on emulators or virtual machines, while others execute directly on host machines for better efficiency.

2.1 Unpackers on Controlled Environments

A controlled environment may use emulators (e.g., Qemu, Bochs) or virtual machines (e.g., VMware, VirtualPC). Unpackers on controlled environments first dynamically execute the packed code for a period of time that is sufficient for completing the unpacking, and then extract the real payload. It is usually tricky to choose the time period which is sufficient to extract the actual payload. If the malware payload gets executed within this time period, the executing environment of the unpacker runs into an unsafe mode. However, this is not an issue as the controlled environment guarantees the safety of the host machine. Even if the emulated or virtual machine is infected with viruses, it can be restored by, e.g., previous and clean snapshots.

Examples in this category include PolyUnpack [14], which compares the executing instructions with static disassembly code by monitoring the program's execution. Malware Normalizer [2], Renovo [8], Azure [13], Saffron [11], Paradyn [2], and Pandora [1] unpack the program based on the fact that the packed code has to write the instruction in memory and then execute these instructions. Renovo and Pandora use emulators to monitor the memory access, Saffron and Paradyn use dynamic instrumentation, and Azure uses Intel's VT extension. Eureka [15] monitors two system calls, `NtTerminateProcess` and `NtCreateProcess`, to find the malicious payload.

Using emulators and virtual machines have two challenges in general. One is that running a program in emulators or virtual machines is much slower (up to several hundreds of times [17]) than running it on a host machine. Another is due to the various anti-emulation and anti-vm techniques [3,4,12] that may terminate the program's execution once the existence of a controlled environment is detected.

² Paradyn, <http://pages.cs.wisc.edu/~paradyn/>.

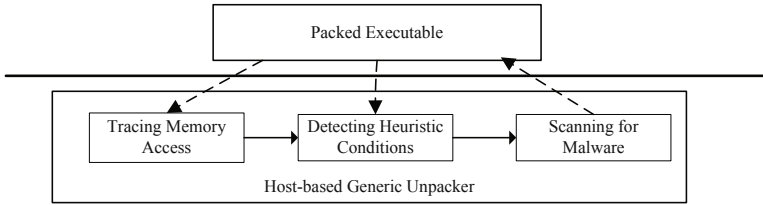


Fig. 1. Host-based unpackers

2.2 Host-Based Unpackers

Running the unpacker on a host machine improves the performance but might leave the system in an unsafe state. Therefore, it is important for unpackers to terminate the execution once the unpacking routine is completed and before the malicious payload gets executed. Heuristics are introduced to approximate the end of unpacking. Fig. 1 gives an overview of host-based generic unpackers.

In general, the unpacker monitors memory accesses of the packed code and detects satisfying conditions of the heuristics. Whenever the conditions are met, the unpacker invokes anti-virus software to scan the (partially) unpacked code to decide the end of unpacking. The calling of the anti-virus software is important as the unpacker needs to protect its operating environment not to be infected by the malware.

Both OmniUnpack [9] and Justin [5] monitor events when a memory page is written and subsequently executed, although they use different heuristics. OmniUnpack uses a heuristic that when the malware gets executed, it needs to interact with the operating system using dangerous system calls. Justin incorporates other heuristics including unpacker memory avoidance, stack pointer checking, and command-line argument checking. In order to improve performance, both of them monitor memory accesses at the page level.

Note that these heuristics adopted by host-based unpackers might not be always accurate. For example, an invocation of a dangerous system call is not a sufficient condition to indicate end of the unpacking or execution of the malware.

2.3 Overhead of Host-Based Unpackers

We used our implementation of a previously proposed unpacker (see Section 4) to unpack a number of packed malware (Amanda, BirdWatcher, Aimbot, Arus, BlackEyes, Adroar, Aidid and DenisBee) and analyzed the breakdown of the time spent. Fig. 2 shows the average time spent in each stage of the unpacking. It shows that the time taken by the anti-virus software dominates the overhead of the unpacker. This result is consistent with another report on the time spent by anti-virus software [6], in which it was reported that scanning a binary or system file of 1 Mbytes takes at least 0.02 seconds for more than 40 AV-scanners (the average size of the malware we tested is 61 Kbytes, which corresponds to 0.01953 second per Mbytes).

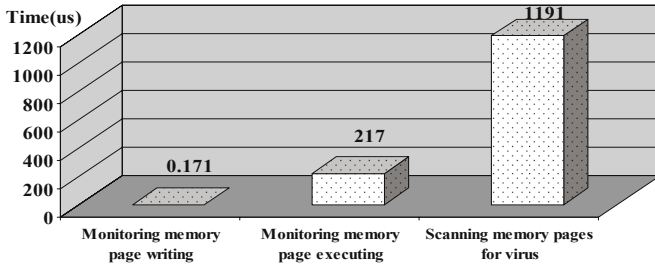


Fig. 2. Overhead of host-based unpackers

Inspired by this analysis, we designed packed malware that results in a large number of invocations of the anti-virus software in order to make effective denial-of-service attacks on host-based unpackers.

3 Denial-of-Service Attack and Its Implementation

Fig. 3 shows an overview of our denial-of-service attack on a particular host-based generic unpacker, OmniUnpack. The packed malware first detects the existence of the unpacker in the executing environment (see Section 3.1), then forges (a loop of) spurious behavior if an unpacker is detected or executes the original malware otherwise. Each iteration of the spurious behavior tries to satisfy the heuristic condition of the unpacker, which, for example, includes writing to a memory page, executing the memory page, and making a dangerous system call.

The spurious behavior satisfying the heuristic conditions of the unpacker will result in an invocation of the anti-virus software to scan for viruses. T_W , T_E , and T_S represent the overhead introduced by monitoring `write` behavior, `execute` behavior, and scanning for malware by the anti-virus software, respectively.

Designing this denial-of-service attack in a packed malware with all the properties shown in Section 1 has a few challenges. First, the packed malware needs to be able to detect the existence of an unpacker. This ensures that the original

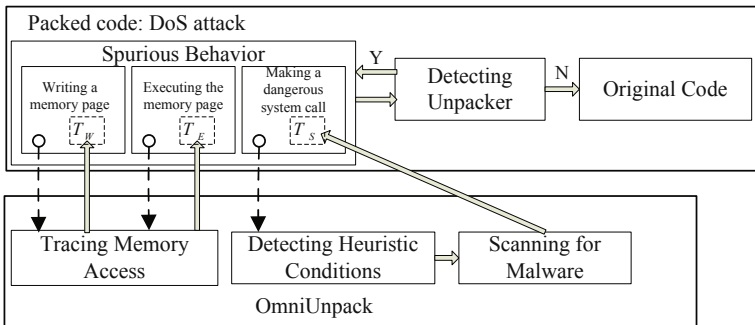


Fig. 3. Overview of the DoS attack on OmniUnpack

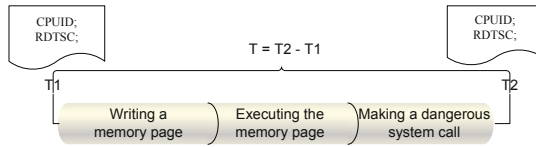


Fig. 4. Measuring CPU cycles to detect the existence of an unpacker

malware will be executed when the packed code is run on a normal machine (the victim of the original malware). Second, the code that facilitates the denial-of-service attack, including the forged spurious behavior, should not be flagged as a virus by the anti-virus software. If it is, it will be detected the first time the anti-virus software is triggered, which in turn will result in the success of the unpacker in detecting the malware, and the denial-of-service attack will fail. On the other hand, as long as this code is benign, the anti-virus software will have to be invoked for every iteration of the spurious behaviors, which forms the base of our denial-of-service attack.

3.1 Detecting the Existence of an Unpacker

Generic unpackers introduce overhead in, e.g., monitoring memory access and invoking anti-virus software. To distinguish between executing environments with and without a generic unpacker, we monitor the time-stamp counter of the processor (TSC) to measure the number of CPU clock cycles in executing spurious unpacking routines. Figure 4 shows the details of such a measurement.

To implement this checking, we used the RDTSC instruction to measure the number of CPU cycles. We measure T (the difference between $T2$ and $T1$) many times (each of which consists of 5 iterations of the execution of a spurious unpacking routine to reduce the cache warm-up effect) to find its upper bound. By applying standard statistical techniques to our experiments, we arrived at a threshold value of 5,000 clock cycles, i.e., if T exceeds 5,000, it is most likely that the process is being executed with an unpacker. Please refer to Section 3.4 for some implementation details.

One possible way for the unpacker to confuse this measurement is to fake the RDTSC instruction by hooking the Interrupt Descriptor Table (IDT)³. However this needs a driver to be executed in supervisor mode (ring 0) and may affect the system stability. Dealing with this issue is beyond the scope of this paper.

3.2 Spurious Unpacking Behavior

As mentioned in Section 2.2, host-based generic unpackers adopt various heuristics to approximate the end of unpacking. These heuristics help identifying the potential points in the execution of the packed code that are likely to be the end of unpacking. However, these potential points in the execution of the packed

³ Fake RDTSC, ARTeam, <http://deroko.phearless.org/ring0.html>

code might not be the correct ones. The host-based generic unpackers understand the potential inaccuracy and invoke the anti-virus software at all these potential points of execution. Our denial-of-service attack forges a large number of the spurious unpacking behaviors so that the unpacker invokes the anti-virus software many times.

To make the discussion more concrete, here we take OmniUnpack as an example. As discussed in Section 2.2, OmniUnpack monitors events when a memory page is written and subsequently executed, and uses a heuristic that when the malware gets executed, it needs to interact with the operating system using dangerous system calls. Our denial-of-service attack first writes a memory page p with predefined instructions which do no meaningful operation (e.g., `nop`), and then executes instructions in p . After that, it invokes a dangerous system call. Note that as discussed at the beginning of Section 3, the forged spurious behavior should not be flagged as a virus by the anti-virus software. To satisfy this requirement, we choose to first create a file and then invoke the dangerous system call `NtDeleteFile` to delete it. We tested this with a number of anti-virus software and found that none of them flags such behavior as malicious.

3.3 Our Automatic Packer

To implement a packer that takes as input unpacked malware and outputs packed code that performs our denial-of-service attack on a host-based generic unpacker, we took a packer from aPLib⁴ and modified its compressor engine. Our modified packer is not very different from the original one in their construction of packed code, i.e., both will first load the original malware binary into memory, compress the data from different sections (e.g., `.text`, `.data` and `.rsrc`), and obfuscate the import table to make reverse engineering difficult, and then add the code needed for unpacking (decompressing the packed code and rebuilding the import table). The difference is that our modified packer also inserts the DoS attack stub, which gets executed before the original binary is decompressed. Fig. 5 presents the structure of the packed code produced by our automatic packer.

At load time the DoS attack stub is first executed to detect whether a host-based generic unpacker exists. If an unpacker does not exist, the code starts decompressing the packed malware and rebuilding the original imports using addresses obtained from APIs `LoadLibrary` and `GetProcAddress`. Control is then transferred to the entry point of the decompressed code, and the original malware binary gets executed. If an unpacker exists, the DoS attack code starts iterations of the spurious behavior.

3.4 Implementation Details

In order to make our measurement of TSC more accurate, we took into consideration several issues. First, because Intel CPU supports out-of-order execution, we flush the instruction pipeline to prevent early termination of the measurement.

⁴ aPLib compression library, <http://www.ibsensoftware.com/>

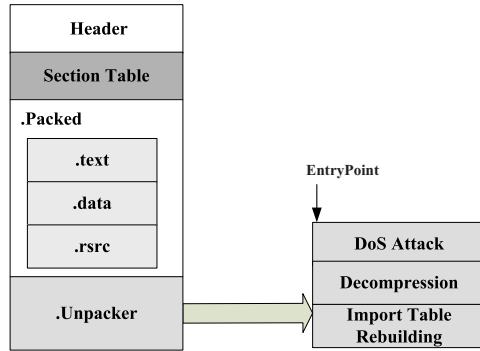


Fig. 5. Structure of the packed code

The `CPUID` instruction is used for serializing execution [7]. We also subtract the overhead of instructions `RDTSC` and `CPUID` from our results (350 to 400 clock cycles). Second, cache effect might bias our results. It takes a large number of cycles to load data and code into the cache when execution starts. To reduce the cache warm-up effect, we repeat the measurement at least five times and discount the first measurement. Third, time-stamp counters on different cores may not be synchronized with each other. Instruction `SetThreadAffinityMask` is used to force execution on only one core.

Context switch and memory page swapping sometimes confuse our detection of the existence of an unpacker, because they introduce big overhead even if an unpacker does not exist. In order to reduce the false-positive rate of detecting an unpacker, we choose to detect the existence of an unpacker every time the spurious unpacking behavior is generated. The overhead introduced by this checking is very small compared to the overhead of the spurious behavior generated, but it lowers the false-positive rate significantly. In our experiments, our packed malware always manages to detect the existence of an unpacker correctly with at most two rounds of spurious behaviors.

4 Evaluation

In order to evaluate our proposed denial-of-service attack and the packer we implemented, we chose one of the latest and most sophisticated host-based generic unpackers, `OmniUnpack` [9], as the target. Since we do not have the `OmniUnpack` source code from the authors, we implemented it ourselves based on the descriptions from the paper and some help from the `OmniUnpack` authors on a computer with an Intel CPU of 3 GHz and 1 Gbytes of memory running Windows XP SP3.

There are three parts in our evaluation. First, we use anti-virus software to scan a few pieces of malware packed using existing packers and our packer. Second, we execute malware packed using our packer on normal computers without

Table 1. Size of the packed code (bytes)

	Amanda	BirdWatcher	Aimbot	Arus	BlackEyes	Adroar	Aidid	DenisBee
Original packer	40,960	77,824	20,480	186,368	20,480	77,824	20,480	41,472
Our packer	41,472	78,336	20,992	186,880	20,992	78,336	20,992	41,984

Table 2. Packed malware escaping detection of anti-virus software

	Amanda	BirdWatcher	Aimbot	Arus	BlackEyes	Adroar	Aidid	DenisBee
Kaspersky								
ClamAV								
Mcafee	⊗	⊗	⊗			⊗ ⊗	⊗ ⊗	⊗ ⊗
Microsoft Protection	⊗				⊗		⊗	⊗
NOD32	⊗	⊗ ⊗	⊗			⊗	⊗	⊗
Norman		⊗						
TrendMicro					⊗ ⊗			
nProtect	⊗ ⊗	⊗	⊗ ⊗		⊗ ⊗			

⊗: malware packed with the original packer is detected

⊗: malware packed with our packer is detected

unpackers. Third, we execute malware packed using our packer on machines with our implementation of the OmniUnpack.

4.1 Anti-virus Software Detection

This part of the evaluation tries to see if the code added by our packer can escape detection by various anti-virus software. As described in Section 3.3, our packer is implemented by adding our denial-of-service attack code to an existing packer from aPLib. We first measure the size of the packed code with the original packer from aPLib and the one with our packer. Results are shown in Table 1. We find that our packer adds roughly 0.843% overhead compared to packed code with the original packer.

In order to find whether the original packer from aPLib or our denial-of-service attack code contributes to anti-virus detection, we use anti-virus software to scan both the malware packed using the original packer from aPLib and packed using our packer. Table 2 shows the result.

Columns and rows in Table 2 show different malware samples and anti-virus software we have tested, respectively. Results show that the denial-of-service attack code introduced by our packer does not contribute to the detection by anti-virus software, because in all the eight cases where malware packed with our packer is detected, the same malware packed using the original packer is also detected. Also note that there are a few cases where malware packed using the original packer is detected, while the one packed with ours is not.

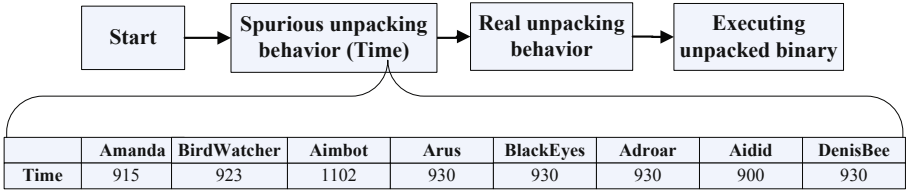


Fig. 6. Malware execution without host-based generic unpackers

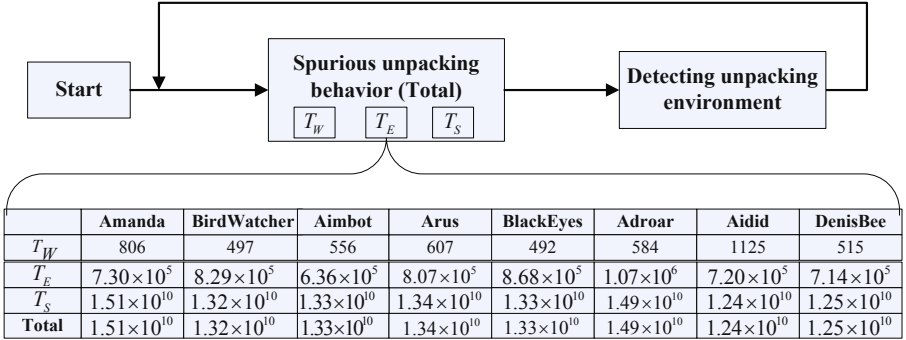


Fig. 7. Malware execution with OmniUnpack

4.2 Executing Our Packed Malware

In this part of the evaluation, we executed malware packed using our packer on two different environments, one with an unpacker (our implementation of OmniUnpack) and one without an unpacker (victim of the malware). We used the same anti-virus software OmniUnpack uses, i.e., ClamAV⁵. OmniUnpack claims that it only scans modified pages since the last dangerous system call. However according to a later report [5], this scanning strategy is not compatible with most existing commercial AV-scanners. Instead, we dumped the whole memory image and scan it with ClamAV just as what Justin [5] does. We also instrumented our implementation of the OmniUnpack to record T_W , T_E , and T_S in every spurious unpacking behavior found.

We ran eight malware samples (the same as those reported in Table 2) packed with our packer. Fig. 6 and Fig. 7 show our observations of the execution of these eight malware samples with and without OmniUnpack in the executing environment, respectively. Numbers shown in the figures are number of CPU cycles. We found that all eight packed malware samples succeeded in the denial-of-service attack on OmniUnpack by forging a large number of spurious unpacking behaviors (Fig. 7). At the same time, they all managed to unpack the original malware with only five spurious unpacking iterations on a normal executing environment

⁵ Clam AntiVirus, <http://www.clamav.net/>

(Fig. 6 only shows the last spurious iteration, and the first four are used to reduce the cache warm-up effect, see Section 3.1).

A closer look at Fig. 6 reveals that only about 1,000 CPU cycles were spent in the first spurious unpacking iteration, which is well below the threshold used in detecting the executing unpacking environment (see Section 3.1). However, this number is several orders of magnitude larger in the presence of OmniUnpack (see Fig. 6). This result verifies that our packed malware detects the existence of a host-based generic unpacker correctly.

4.3 Applicability of Our Attack and Limitations

Unpackers like OmniUnpack could introduce a counter to monitor the number of spurious unpacking behaviors and use a threshold to fight against our denial-of-service attack. However, this does not solve the problem as it is hard to set such a threshold and it comes with the price of failure in unpacking many packed code. This is because the idea of our attack is based on 1) end of unpacking is undecidable and heuristics have to be used for approximation; and 2) host-based generic unpackers need to protect itself by invoking anti-virus software in all potential execution points of the end of unpacking. We do not believe a simple solution exists for practical host-based generic unpackers in fighting against our denial-of-service attack.

However, our attack is not applicable to the unpackers which adopt very different heuristics. For example, Eureka [15] introduces statistical n-gram analysis to help find the end of unpacking. Hump-and-dump [18] creates a histogram of the addresses of executed instructions ordered by the last time the address is executed. Our attack will not work on these unpackers.

5 Related Work

Ferrie introduced various anti-unpacking tricks, including anti-dumping, anti-debugging, anti-emulating, and anti-intercepting techniques [4]. Since the real payload will eventually be unpacked to the memory, unpackers dump the memory when they detect the existence of the payload. Anti-dumping techniques are designed to prevent the unpacker from dumping the process memory. Several unpackers make use of a debugger or emulator to monitor the unpacking behavior. Anti-debugging and anti-emulating tricks are used to circumvent the unpacking whenever there is a debugger or an emulator running. Some unpackers use page interceptor to monitor memory page accesses. Anti-intercepting techniques are used to detect the existence of an interceptor and evade it.

There have also been works to detect the existence of an emulator [3, 12], including CPU bugs, model-specific registers, and alignment checking. These techniques can be adopted to evade emulator-based unpackers.

Dual-mapping is used to evade unpackers which dynamically unpack the program by detecting memory execution [16]. In this work, physical memory regions are mapped into two virtual ones, one for written and another for execution.

Our work is different from these previous work in that we use timing information for detecting the existence of an unpacker.

6 Conclusion

In this paper, we present a denial-of-service attack on host-based generic unpackers. Malware samples packed using our automatic packer detects the existence of a host-based generic unpacker and forges spurious unpacking behaviors, and these spurious unpacking behaviors result in a denial-of-service attack on the unpacker. Our experimental evaluation demonstrates the effectiveness of this attack.

Acknowledgements

We thank the authors of OmniUnpack for their help in our implementation and sharing of information in the dangerous system call table.

References

1. Bohne, L.: Pandora's Bochs: Automatic Unpacking of Malware. PhD thesis, University of Mannheim (2008)
2. Christodorescu, M., Kinder, J., Jha, S., Katzenbeisser, S., Veith, H.: Malware normalization. Technical report 1539, University of Wisconsin, Madison, WI, USA (2005)
3. Ferrie, P.: Attacks on virtual machine emulation. In: AVAR Conference. Symantec Advanced Threat Research (2006)
4. Ferrie, P.: Anti-unpacker tricks. In: Proceedings of the 2nd International CARO Workshop (2008)
5. Guo, F., Ferrie, P., Chiueh, T.C.: A study of the packer problem and its solutions. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 98–115. Springer, Heidelberg (2008)
6. Hawes, J.: Comparative review. Virus Bulletin, 14–27 (2009)
7. Intel Corporation. Using the RDTSC Instruction for Performance Monitoring (1997)
8. Kang, M.G., Poosankam, P., Yin, H.: Renovo: A hidden code extractor for packed executables. In: Proceedings of the 2007 ACM workshop on Recurring malware (WORM), pp. 46–53 (2007)
9. Martignoni, L., Christodorescu, M., Jha, S.: Omniunpack: Fast, generic, and safe unpacking of malware. In: Proceedings of the 2007 Annual Computer Security Applications Conference (ACSAC), pp. 431–441. IEEE Computer Society, Los Alamitos (2007)
10. Morgenstern, M., Marx, A.: Runtime packer testing experiences. In: Proceedings of the 2nd International CARO Workshop (2008)
11. Quist, D.: Covert debugging: Circumventing software armoring techniques. In: Black Hat (2007)
12. Raffetseder, T., Krügel, C., Kirda, E.: Detecting system emulators. In: Proceedings of 10th International Conference on Information Security (ISC), pp. 1–18 (2007)
13. Royal, P.: Alternative medicine: The malware analyst's blue pill. In: Black Hat (2008)
14. Royal, P., Halpin, M., Dagon, D., Edmonds, R., Lee, W.: Polyunpack: Automating the hidden-code extraction of unpack-executing malware. In: Proceedings of 2006 Annual Computer Security Applications Conference (ACSAC), pp. 289–300. IEEE Computer Society, Los Alamitos (2006)

15. Sharif, M., Yegneswaran, V., Saidi, H., Porras, P., Lee, W.: Eureka: A framework for enabling static malware analysis. In: Proceedings of 13th European Symposium on Research in Computer Security (ESORICS), pp. 481–500 (2008)
16. Skape: Using dual-mappings to evade automated unpackers. *Uninformed Journal* (2008)
17. Stepan, A.: Improving proactive detection of packed malware. *Virus Bulletin*, 11–13 (2006)
18. Sun, L., Ebringer, T., Boztas, S.: Hump-and-dump: Efficient generic unpacking using an ordered address execution histogram. In: Second International CARO Workshop, Department of Computer Science and Software Engineering, The University of Melbourne, Australia (2008)

Predictive Pattern Matching for Scalable Network Intrusion Detection

Lucas Vespa, Mini Mathew, and Ning Weng

Department of Electrical and Computer Engineering
Southern Illinois University Carbondale, Carbondale IL 62901, USA
{lvespa,mini10,nweng}@siu.edu

Abstract. Signature-based network intrusion detection requires fast and reconfigurable pattern matching for deep packet inspection. This paper presents a novel pattern matching engine, which exploits a memory-based programmable state machine to achieve deterministic processing rates that are independent of packet and pattern characteristics. Our engine is a portable predictive pattern matching finite state machine (P^3FSM), which combines the properties of hardware-based systems with the portability and programmability of software. Specifically we introduce two methods, “Character Aware” and “SDFA”, for encoding predictive state codes which can forecast the next states of our FSM. The result is software based pattern matching which is fast, reconfigurable, memory-efficient and portable.

1 Introduction

Network Intrusion Detection System (NIDS) is one of the most important aspects of any computer network [4,8]. Specifically, signature-based NIDS provides the comprehensive detailed detection capabilities that are vital to preventing network attacks [5]. The key challenge with the deep packet inspection performed by signature-based NIDS, is that it requires fast, programmable and efficient pattern matching capabilities [2,6] that have yet to be developed in a well rounded solution.

Previous attempts to solve this problem include pattern matching engines that can be hardware or software based. Hardware-based pattern matching can achieve acceptable speed, programmability and memory efficiency [3,11], however, these solutions require specialized hardware in order to implement them. Software-based pattern matching is universally portable, however still lacks the sheer speed and deterministic properties necessary to keep up with network line rates [10].

Our goal in this work is to develop a well-rounded solution that combines the properties of hardware-based systems with the portability of software-based systems, to produce a software NIDS that is fast, programmable and memory-efficient, solving the overall problem of signature-based NIDS. This system will fill the gap between highly programmable, portable, yet slow software based NIDS; and fast, memory efficient, yet hard to implement, hardware based NIDS. Works like [9] which is a hardware-based system have exploited the deterministic nature of Deterministic Finite Automata (DFA), and created solutions to the

excessive memory requirements of DFA. It is these concepts, which are built upon and extended by our design called P^3FSM .

P^3FSM is a software-based Finite State Machine that implements Deterministic Finite Automata using predictive state codes. DFA are used because of their deterministic behavior, which allows for fast, deterministic packet processing, regardless of the number and size of patterns. DFA however require excessive amounts of memory to be stored. P^3FSM solves this problem by storing only a single code for each state in a DFA. These codes are derived such that each state code can forecast all of its possible next states. An FSM is formed from these state codes which allows for quick system operation. The result is a pattern matching engine which combines the benefits of both hardware and software systems. P^3FSM is fast, memory-efficient, highly programmable and still completely portable. Several different implementations of P^3FSM , along with experimental results are presented in order to validate its operation as well as demonstrate its many benefits.

The remainder of this work is organized as follows. Related work is discussed in Section 2. The overall system architecture is discussed in Section 3, and Sections 4, 5 and 6 demonstrate the programming and operation of P^3FSM . Experimental results are discussed in Section 7 and Section 8 concludes the paper.

2 Related Work

A flurry of work has been proposed to design a high performance string matching engine. However, few works consider memory efficiency and ease of update of new patterns. We present some background and most related work to P^3FSM .

2.1 Deterministic Finite Automata

Deterministic Finite Automata (DFA) are able to match multiple strings simultaneously, in *worst-case* time linear to the size of a packet. Figure 2 shows an example DFA used to match “SHE”, “HERS” and “HIS”. Starting at state s_0 , the state machine is traversed to state s_1 or s_2 depending on whether the input character is “H” or “S”. When an end state is reached, a string has been said to be matched. In the example in Figure 2, if state s_9 is reached, string “HERS” has been matched.

Each state in the machine has pointers to other states in the machine. If an input character is the next character in a string that is currently being matched, the algorithm moves to the next state in that string, otherwise, the algorithm follows a failure pointer to the first state of another string that begins with that character, or to the initial state of the machine if no other strings begin with that character. An example of this can be seen in Figure 2. If the current state of the machine is s_3 , the last input characters would have been “HE”. If the next input character were to be “R”, then the next state would be s_6 . If the next input character were not “R”, but instead, “S”, then the next state would follow a failure pointer to state s_2 , which is the starting point for the string “SHE”.

2.2 Storage Requirement of Traditional DFA

A DFA can be implemented using hardware or memory. In a memory based implementation of a DFA, the Current State and Input Character are used as the memory address location of the memory content. The memory content consists of the Next State and Tag. A memory implementation of a DFA can be easily reconfigured by reprogramming the memory with a new or updated State Transition Table.

In a memory based implementation of a DFA, the memory size needed to hold the State Transition Table is based on the number of bits needed to represent each state s and the number of bits needed to represent each input character (8 bit). For example, Snort Dec. 05 has 2733 patterns, which need 27,000 states to represent them so the storage requirement is about 13MB, which is too big to fit onto on-chip memory.

The amount of memory needed to store a DFA is large and increases greatly as the number of strings being matched increases. The reason for the large memory requirement in traditional memory-based FSM is that all possible 256 next states of any given state are explicitly stored in the memory array even though many of these next states are the same.

2.3 Memory Efficiency Optimization

The Aho-Corasick (AC) algorithm [11] is able to match multiple strings simultaneously by constructing a state machine. Starting from an empty root state, each string to be matched is represented by a series of states in the machine, along with pointers to the next appropriate state. A pointer is added from each node to the longest prefix of that node which also leads to a valid node in the machine. The major drawback of the AC algorithm is possible 256 fan-out, which results in low memory efficiency.

Bitmap and compression [12] have been proposed to optimize the AC algorithm data structure to improve the memory efficiency. The problems of bitmap compression require 2 memory references per character in the worst case and 256 bits per bitmap. A potential problem with path compression is, failure pointers may point to the middle of other path compressed nodes.

Our recent work [9] presents the idea of using a novel encoding method to implement a DFA with an FSM that stores only one entry in memory for each state, rather than an entry for each transition. The problem of our previous work [9] has very limited portability and flexibility due to the hardware-coded group detector. Our P^3FSM addresses this limitation by a software detection engine, which leverages packet pre-fetching and character-aware encoding to achieve comparative performance as hardware.

Luntenen [6] introduces a pattern matching engine called BFPM, or Balance Randomized Tree FSM Pattern Matching. This approach is similar to our approach in the way it utilizes DFA. BFPM uses a prioritized rule list which reduces the number of transitions in a DFA. This is similar to the “SDFA” optimization discussed in Section 3.

3 System Architecture

P^3FSM is a fast, space-efficient, fully-programable and portable pattern-matching engine which runs purely in software. P^3FSM is unique in the way that it mimics the very desirable properties of DFA based pattern matching hardware, but is completely software-based. The system utilizes a DFA in order to maintain deterministic performance, however, the DFA is stored in a novel way such that minimum memory is required. This means that secondary memory is not required in P^3FSM making both memory-efficiency and performance excellent. P^3FSM achieves these properties by only storing one entry in memory for each state in a DFA, rather than storing numerous DFA transitions as in typical memory based DFA implementations. The code for each state is derived such that it is predictive, in that, each state code contains information that denotes all of its possible next states. The FSM formed from these codes is able to isolate the appropriate next state very quickly due to the unique properties of these codes. Multiple algorithms for deriving state codes and for FSM operation are discussed in Sections 4, 5 and 6.

The overall functionality of P^3FSM software comes from two distinct components, the Pattern Compiler and FSM. This architecture is illustrated by Figure 1. The Pattern Compiler can further be broken down into two components, Pattern Analysis and State Analysis. In the Pattern Analysis component, the first step is the formation of a DFA from patterns. A clustering algorithm is then used to generate state codes. The DFA and state codes are then passed to the State Analysis component. This is where all the information needed for FSM functionality is organized. Select information about characters, clusters (each state code contains clusters which is discussed later) and state codes is stored in tables which are where the FSM Operation component gets the information

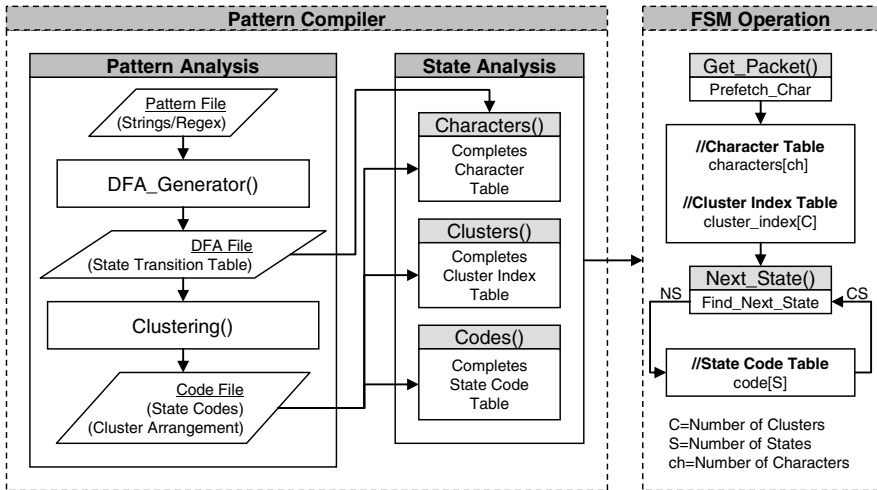


Fig. 1. System Architecture

needed to operate. After the Pattern Compiler completes, the FSM component can operate. The FSM Operation component is responsible for the actual pattern matching of P^3FSM .

As aforementioned, the following sections contain examples of different P^3FSM approaches. The names for the different approaches are derived from the way in which the state codes for the FSM are generated. The state encoding algorithm used in any approach may utilize a property called “Character Aware”. When deriving state codes from a DFA, states are grouped and clustered together in order to generate self-addressable state codes for each state. During the clustering process, if the algorithm takes into account the character required to transition to each state, it is a “Character Aware” algorithm. Second, if the DFA used to derive the state codes goes through an optimization which we have deemed Split-DFA (SDFA), then it is an “SDFA” algorithm. All of the approaches discussed in this paper utilize at least one of these properties, either “Character-Aware” or “SDFA”, and the algorithms are named based on the combination of these properties that they utilize. Depending on which properties are used, the resulting FSM will have either smaller memory requirements, greater throughput or a trade-off between the two.

4 Approach A: “Character Aware” for High-Throughput

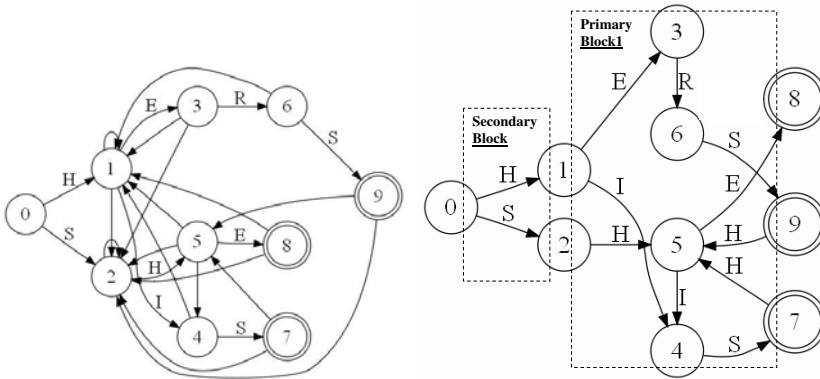
Discussed here is the first implementation of P^3FSM . This approach is called “Character Aware”. What can be observed about this approach is that because it is “Character Aware” it has excellent throughput, more-so than the other approaches. However, it is not as memory efficient. More about this will be discussed in the Results section. The following sections describe by example the different components, starting with the Pattern Compiler and then the operation of the FSM.

4.1 Pattern Compiler: Pattern Analysis

This section describes the Pattern Analysis component of the Pattern Compiler. The two stages of this component are construction of a DFA and the derivation of state codes from the DFA.

Deriving a DFA. The first step of the Pattern Analysis component is to construct a DFA from a set of patterns. Figure 2a shows a DFA constructed from the patterns “SHE”, “HERS” and “HIS”.

Deriving State Codes. The second step of the Pattern Analysis component is to derive the predictive state codes for each state. The following example illustrates the derivation of state codes for the DFA in Figure 2b. The first step in deriving the state codes is to group all the states in the DFA that have the same next state into a group. The result is one group for each state (ie. G1 = the fan-in of S1), etc. Also, the character needed to transition to a state is associated with that state’s group (ie. S1.character = ‘H’). The resulting groups are: G1[S0, S1, S3, S4, S5,S6,S8][H], G2[S0, S1, S2, S3, S5, S7, S8, S9][S], G3[S1][E], G4[S1, S5][I], G5[S2, S7, S9][H], G6[S3][R], G7[S4][S], G8[S5][E], G9[S6][S].



(a) DFA for Patterns “SHE”, “HERS”, “HIS” (b) S DFA with Primary and Secondary Blocks

Fig. 2. DFA before and after S DFA Modification

Next clusters are formed by placing all the groups with same character into one cluster as shown in Table 1. The next step of this process is generating a code for each group. The group code for every character begins with 1 and increments for each group, starting over at the beginning of each cluster. The group codes are given in Table 2. The code for any group is referred to as the state signature of the state which the group is a representative of.

Finally, a state code for each state is obtained by concatenating the group codes for the groups that a state is a member of. The group codes are formed into a state code by being placed in the position of the cluster that the group is a member of. Table 3 shows the state codes for all states. As an example, state S4 is a member of groups G1 and G7 and the codes for G1 and G7 are 01 and 10 respectively. The state code for S4 is formed by placing the code for G1 in the position of cluster C1 and the code for G7 in the position of cluster C2. The remaining cluster positions receive all zeros as state S4 is not a member of a group that lies in clusters C3, C4 or C5. The state code for S4 is thusly a concatenation of 01, 10, 00, 0, and 0 which yields the final code 01100000.

4.2 Pattern Compiler: State Analysis

This section describes the State Analysis component of the Pattern Compiler. This consists of the creation of the tables involved in FSM Operation.

Character/Cluster Table. The Character/Cluster Table (Table 4) contains the cluster index required to derive the next state. The index starts at 0 for cluster 1 and is incremented for each cluster by the number of group members in the previous cluster. The character for each cluster is also included in the table.

Code Table. The calculation for deriving the index for any state, which determines its order in the Code Table, is seen in Equation 1. The state index is simply

Table 1. Clustering (A)

Cluster	Character	Group	Bit Length
C1	H	G1 G5	2
C2	S	G2 G7 G9	2
C3	E	G3 G8	2
C4	I	G4	1
C5	R	G6	1

Table 2. Group Coding (A)

Group	Coding
G1	0 1
G5	1 0
G2	0 1
G7	1 0
G9	1 1
G3	0 1
G8	1 0
G4	1
G6	1

Table 3. State Codes (A)

State	Group	Code
S0	G1 G2	0 1 0 1 0 0 0 0
S1	G1 G2 G3 G4	0 1 0 1 0 1 1 0
S2	G2 G5	1 0 0 1 0 0 0 0
S3	G1 G2 G6	0 1 0 1 0 0 0 1
S4	G1 G7	0 1 1 0 0 0 0 0
S5	G1 G2 G4 G8	0 1 0 1 1 0 1 0
S6	G1 G9	0 1 1 1 0 0 0 0
S7	G2 G5	1 0 0 1 0 0 0 0
S8	G1 G2	0 1 0 1 0 0 0 0
S9	G2 G5	1 0 0 1 0 0 0 0

Table 4. Character/Cluster Table (A)

Character	Cluster	Index
H	1	0
S	2	2
E	3	5
I	4	7
R	5	8

Table 5. Code Table (A)

Index	State Code	State
0	0 1 0 1 0 0 0 0	S0
1	0 1 0 1 0 1 1 0	S1
2	0 1 0 1 1 0 1 0	S5
3	1 0 0 1 0 0 0 0	S2
4	1 0 0 1 0 0 0 0	S7
5	1 0 0 1 0 0 0 0	S9
6	0 1 0 1 0 0 0 1	S3
7	0 1 0 1 0 0 0 0	S8
8	0 1 1 0 0 0 0 0	S4
9	0 1 1 1 0 0 0 0	S6

the sum of the cluster index for a state and the value of its state signature. Table 5 shows the completed code table for this example. The state codes are stored in this table in the order of their index which is calculated as aforementioned.

$$S_{index} = Cl_{index} + S_{sig} \tag{1}$$

4.3 FSM Operation

The following example explains the functionality of the FSM Operation component. Consider the current state as S3 and incoming character ‘R’. To derive the next state for incoming character ‘R’ we require the cluster index and the state signature. Table 4 shows that ‘R’ belongs to cluster 5. We obtain the cluster index for cluster 5 from Table 4 which is 8. The state signature is found in cluster 5 of the current code S3. From the Code Table the code for S3 is 01010001. This means that the state signature for cluster 5 in S3 is 1. Using Equation 1 we get the next state index which is calculated as, $8 + 1 = 9$. Therefore, the index 9 in the Code Table gives the next state as S6.

5 Approach B: “SDFA” for Reduced Memory

Discussed here is the second implementation of P^3FSM . This approach is called “SDFA”. What can be observed about this approach is that because it uses the “SDFA” optimization its memory requirements are very small, more-so than the previous approach. However, it has a much lower throughput. More about this will be discussed in the Results section.

5.1 Pattern Compiler

For this approach, after the DFA is formed as aforementioned, it is optimized into what we have deemed a Split-DFA or SDFA. SDFA eliminates redundant transitions and then partitions the DFA into multiple blocks called the Primary-Block and Secondary-Block. States which have a high-density of redundant transitions are those which also receive transitions from the initial or zero state of the DFA. We call these states first-level states as they hierarchically lie on the first level of the DFA. All transitions to these states other than those from state zero are removed from the DFA and subsequently these remaining transitions to first-level states are partitioned into the Secondary-Block of the SDFA as seen in Figure 2b. At this point the majority of transitions have been removed from the DFA (especially in larger DFA), and the remainder of the transitions are said to lie in the Primary-Block of the SDFA.

Groups are formed from states in the DFA as in previous examples, however, this approach does not utilize the “Character Aware” property. This means that groups can be clustered together if they do not share any members, regardless of what characters those groups possess. A maximal clique algorithm is used to perform this clustering operation and the result is the shortest state codes of all three implementations. The state codes derived for this approach can be seen in Table 6.

Table 6. State Codes (B)

State	State Code
S1	0 0 1 1 1
S2	0 1 0 0 0
S3	0 1 0 1 0
S4	0 1 1 0 0
S5	0 1 1 1 1
S6	1 0 0 0 0
S7	0 1 0 0 0
S8	0 0 0 0 0
S9	0 1 0 0 0

5.2 FSM Operation

The FSM Operation for this approach uses several bitwise masking operations to isolate the cluster containing the next state signature. This algorithm has a much greater time complexity than the other two algorithms. The reason for this time complexity is the very short state codes. Although more memory efficient than the other two approaches, these codes tell little about the potential next states and therefore require more work from the FSM to extract the appropriate next state.

6 Approach C: “Character Aware / S DFA” for High Throughput and Reduced Memory

Discussed here is the final implementation of P^3FSM . This approach is called “Character Aware / S DFA”. What can be observed about this approach is that, because it is “Character Aware” it can quickly isolate the next state, however, because of the “S DFA” optimization it is memory efficient. This implementation is the most balanced as far as memory efficiency and throughput. More about this will be discussed in the Results section. The following sections describe by example the different components, starting with the Pattern Compiler and then the FSM component operation.

6.1 Pattern Compiler: Pattern Analysis

This section describes the Pattern Analysis component of the Pattern Compiler.

Deriving and Optimizing a DFA. In this example, state codes are derived from the S DFA in Figure 2b.

Deriving State Codes. The next step which is the derivation of predictive state codes is explained by the following example. The DFA to be referred is given in Figure 2b. All the states in the DFA are grouped together as explained in Section 4.1. The resulting groups are: G1[S0][H], G2[S0][S], G3[S1][E], G4[S1, S5][I], G5[S2, S7, S9][H], G6[S3][R], G7[S4][S], G8[S5][E], G9[S6][S].

The second step is to combine these groups together to form clusters. For this, first the groups with the same character are combined into a cluster. Next, the number of clusters are reduced by merging all the clusters that do not have common states to form one cluster. The clustering result thusly obtained is given in Table 7. As seen in this table, characters ‘H’, ‘S’, ‘E’, ‘R’ do not have common states and hence, can be placed in one cluster.

The third step of this process is encoding the groups. This has two parts, the character signature and state signature. The character signature identifies if the incoming character has a valid state transition (explained in FSM operation) and it begins with 0. This signature remains unchanged for the groups with same character. The state signature gives the next state and always begins with 1. The resulting group codes are shown in Table 8. For example, from the table we see that the groups G1 and G5 have same character signature 00, this is because both the groups have character ‘H’ however their state signatures are 01 and 10 respectively. Therefore, we get the code for G1 as 0001.

The last step is to obtain the state codes as shown in Table 9. The process for constructing state codes from the group codes is the same as in Approach A.

6.2 Pattern Compiler: State Analysis

This section describes the State Analysis component of the Pattern Compiler. This consists of creating tables for use in the FSM operation. Tables for characters, clusters and state codes are created.

Character/Cluster Table. The character/cluster table is created to contain several pieces of information. Firstly, the character signature for each character is stored. Secondly, the cluster that contains all the states associated with that

Table 7. Clustering (C)

Cluster	Character	Group	Bit Length
C1	H S E R	G1 G5 G2 G7 G9 G3 G8 G6	4
C2	I	G4	1

Table 8. Group Coding (C)

Group	Char Sig	State Sig
G1	0 0	0 1
G5	0 0	1 0
G2	0 1	0 1
G7	0 1	1 0
G9	0 1	1 1
G3	1 0	0 1
G8	1 0	1 0
G6	1 1	0 1
G4	0	1

Table 9. State Codes (C)

State	Group	Code
S1	G3 G4	1 0 0 1 0 1
S2	G5	0 0 1 0 0 0
S3	G6	1 1 0 1 0 0
S4	G7	0 1 1 0 0 0
S5	G8 G4	1 0 1 0 0 1
S6	G9	0 1 1 1 0 0
S7	G5	0 0 1 0 0 0
S8		0 0 0 0 0 0
S9	G5	0 0 1 0 0 0

character. Thirdly, each character is given an index number. The index starts at 0 for the first character and is incremented for each character by the number of states with the previous character. Finally, each character is assigned a failure index. This is necessary because of the S DFA optimization done in the pattern analysis component. If a valid transition is not produced during FSM operation, this failure index automatically becomes the next state index as discussed in Section 6.3. Table 10 shows the final Character/Cluster Table for this example.

Code Table. The shared tables used to operate the FSM are the character/cluster table and the code table. The code table consists solely of the state codes placed in the correct order. The index position for each state code in the code table can be calculated by Equation 2, by adding the state signature to the character index for any given state. Table 11 shows the final Code Table for this example.

$$S_{index} = Ch_{index} + S_{sig} \tag{2}$$

6.3 FSM Operation

The functionality of the FSM Operation component is illustrated in the following example. Assume that the current state of the system is S1 and the incoming character is ‘E’. Table 10 shows that this means the current state index is 1. The state code for index 1 is 100101. The character ‘E’ has a signature of 10 and a cluster of 1. Since the character signature in cluster 1 of the current code is also

Table 10. Character/Cluster Table (C)

Character	Signature	Cluster	Index	Failure Index
H	0 0	1	0	1
S	0 1	1	2	3
E	1 0	1	5	0
R	1 1	1	7	0
I	0	2	8	0

Table 11. Code Table (C)

Index	State Code	State
1	1 0 0 1 0 1	S1
2	1 0 1 0 0 1	S5
3	0 0 1 0 0 0	S2
4	0 0 1 0 0 0	S7
5	0 0 1 0 0 0	S9
6	1 1 0 1 0 0	S3
7	0 0 0 0 0 0	S8
8	0 1 1 1 0 0	S6
9	0 1 1 0 0 0	S4

10, then the state signature in cluster 1 which is 01, along with the character index for ‘E’ which is 5 can be used to calculate the next state index. This is done by utilizing Equation 2 with a simple addition: $1 + 5 = 6$. Thus, the next state index is 6 which is the state code for S3. From the DFA in Figure 2b it can be observed that for state S1 on the occurrence of an ‘E’, the next state should be S3, which verifies the next state result.

In a different scenario, assume that the current state is S6 and the incoming character is ‘H’. Referring to Table 11 we get the current state index to be 8 and the corresponding state code 011100. Also, Table 10 gives the cluster for ‘H’ as 1 and the signature 00. The character signature in cluster 1 for S6 is 01 which is different from the signature for ‘H’. This is an invalid condition. Thusly, the next state index becomes the failure index for character ‘H’ which is 1. This gives the next state as S1.

7 Results

The P^3FSM approaches provide noticeable tradeoffs for memory requirements and throughput. As seen in Figure 3 the “SDFA” approach has lowest memory requirement. This is because it removes all the failure transitions as discussed earlier and hence has large cluster sizes yielding small state codes. However, this decreases the throughput drastically because now we may have to perform lookups in different clusters to obtain the next states for a given character. On the other hand, the “Character Aware” approach achieves the highest throughput. This approach stores each character in a separate cluster which increases the speed of lookup as all the possible next states for a given character are now in one cluster. But it also increases the memory requirement because the clustering method used in this approach increases the number of clusters. “Character Aware / SDFA” utilizes the properties of both “SDFA” and “Character Aware”

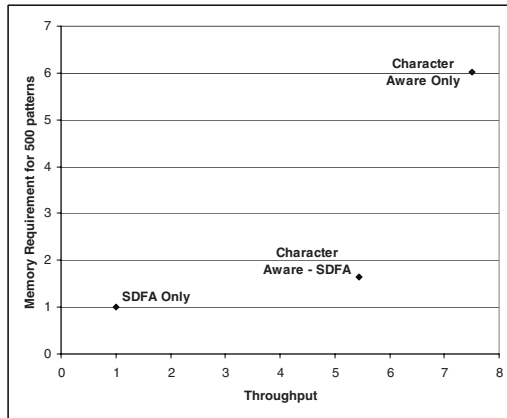


Fig. 3. Memory Requirements vs. Throughput with SDFA Only as Baseline

approaches to produce good memory requirements and throughput. It places many characters into a single cluster but limits the next states of each character to be in one cluster. The simulated results in the following section verify the validity of these approaches.

7.1 Memory Requirements and Performance

Table 12 shows the memory requirements for both a standard DFA, as well as all three P^3FSM implementations. The “SDFA” approach is the most memory efficient, followed by the “Character Aware / SDFA”, and lastly the “Character Aware”. All three approaches are however, more memory efficient than a standard DFA. For 200 patterns, the three P^3FSM implementations range from 10 to 85 times more memory efficient than a standard DFA.

Table 13 contains the throughput and number of instructions per algorithm iteration (FSM Cycles) for the three P^3FSM approaches. As expected, the “SDFA Only” approach has the lowest throughput. It requires more instructions and many more memory accesses than the other methods. The “Character Aware / SDFA” achieves a desirable throughput but is not the fastest implementation. “Character Aware Only” achieves the highest throughput. This is because the state codes used in this implementation allow for a quick isolation of the appropriate next state. This comes with the price of greater memory requirements however.

From both Tables 12 and 13 it can be observed that combining the “SDFA” and “Character Aware” properties into one FSM, allows for a very memory efficient yet still high throughput system. The “Character Aware / SDFA” FSM achieves high throughput while still reducing the memory requirements from a standard DFA by many times.

Table 12. Storage Requirement (in KByte)

Patterns	States	DFA Size	SDFA Size	Character Aware Size	Character Aware / SDFA Size
5	93	23.25	0.08	0.69	0.10
20	302	94.38	0.48	5.16	0.77
50	568	195.25	1.11	13.31	2.50
100	1060	397.50	3.36	34.16	8.15
200	1601	600.38	6.84	60.39	16.03
300	3098	1258.56	22.69	218.21	42.73
400	4837	2116.19	41.92	268.66	72.63
500	5267	2304.31	49.51	298.33	81.65

Table 13. Performance (in Throughput and Number of Instructions)

Performance	SDFA	Character Aware	Character Aware / SDFA
Throughput (in Gbps)	< 0.20	1.50	1.09
Number of Instructions	> 97	13	44

8 Conclusion

In this work we have presented P^3FSM , a fast, portable, memory-efficient and scalable pattern matching engine. P^3FSM is completely software based, yet offers the benefits of many hardware-based pattern matching engines. Three approaches to P^3FSM have also been evaluated, presenting multiple options which concentrate on memory efficiency, throughput and the tradeoff between the two. From simulated results it can be concluded that utilizing predictive state codes to represent a DFA, and taking advantage of intelligent optimizations such as “SDFA” and “Character Aware” clustering, a pattern matching FSM can be implemented which is ideal for NIDS and other high speed applications.

Due to the increasing importance of NIDS, as well as ever growing network speeds, P^3FSM is a suitable solution to modern day signature-based NIDS requirements. It is able to perform its duties at high speeds, yet is portable enough to be easily implemented in any network or end user system.

References

1. Baker, Z., Prasanna, V.K.: A methodology for synthesis of efficient intrusion detection systems on FPGAs. In: Proc. of the IEEE Conference on Field-Programmable Custom Computing Machines, Napa, CA, April 2004, pp. 135–144 (2004)
2. Becchi, M., Crowley, P.: An improved algorithm to accelerate regular expression evaluation. In: Proc. of ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Orlando, FL, December 2007. ACM, New York (2007)
3. Brodie, B.C., Taylor, D.E., Cytron, R.K.: A scalable architecture for high-throughput regular-expression pattern matching. SIGARCH Comput. Archit. News 34(2), 191–202 (2006)
4. Denning, D.: An intrusion-detection model. IEEE Transactions on Software Engineering 13(2), 222–232 (1987)
5. Fisk, M., Varghese, G.: Applying fast string matching to intrusion detection, Los Alamos National Lab Report (2002)
6. van Lunteren, J.: High-performance pattern-matching for intrusion detection. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp. 1–13 (2006)
7. Mahajan, A., Soewito, B., Parsi, S.K., Weng, N., Wang, H.: Implementing High-speed String Matching Hardware for Network Intrusion Detection Systems. In: Proc. of Sixteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Monterey, CA (2008)
8. Roesch, M.: Snort – lightweight intrusion detection for networks. In: Proc. of the 13th Systems Administration Conference (1999)
9. Soewito, B., Vespa, L., Mahajan, A., Weng, N., Wang, H.: Self Addressable Memory-based FSM (SAM-FSM): A Scalable Intrusion Detection Engine. IEEE network 23(1), 14–21 (2009)
10. Wu, S., Manber: A fast algorithm for multi-pattern searching. Technical Report TR94-17, Department of Computer Science, University of Arizona (1994)
11. Aho, A., Corasick, M.: Efficient string matching: An aid to bibliographic search. Communications of the ACM 18 (1975)
12. Tuck, N., Sherwood, T., Calder, B., Varghese, G.: Deterministic memory-efficient string matching algorithms for intrusion detection. In: Proc. of the IEEE Infocom Conference, pp. 333–340 (2004)

Deterministic Finite Automata Characterization for Memory-Based Pattern Matching

Lucas Vespa and Ning Weng

Department of Electrical and Computer Engineering
Southern Illinois University Carbondale, Carbondale IL 62901, USA
{lvespa,nweng}@siu.edu

Abstract. In the midst of vastly numbered and quickly growing internet security threats, Network Intrusion Detection System (NIDS) becomes more important to network security every day. Vital to effective NIDS is a multi-pattern matching engine which requires deterministic performance and adaptability to new threats. Memory-based Deterministic Finite Automata (DFA) are ideal for pattern matching but have severe memory requirements that make them difficult to implement. Many previous heuristic techniques have been proposed to reduce memory requirements, however in this paper, we aim to effectively understand the basic relationship between DFA characteristics and memory, in order to create minimal memory DFA implementations. We show what DFA characteristics either cause or reduce memory requirements, as well as how to optimize DFA to exploit those characteristics. Specifically, we introduce the concepts of State Independence and State Irregularity, which are DFA characteristics that can reduce memory waste and allow for memory reuse. Furthermore, we introduce DFA normalization which optimizes DFA to fully exploit these characteristics. Altogether this work serves as a source for how to extract and utilize DFA characteristics to create minimal memory implementations.

1 Introduction

Deterministic Finite Automata (DFA) have been studied extensively for many purposes [1,2,14]. One of the most common uses is for deterministic pattern matching, specifically for signature-based Network Intrusion Detection Systems [3,4,5] which utilize pattern matching to detect network attack signatures [6,7,8]. DFA are preferred for this purpose as they can be used to match many patterns simultaneously and still achieve deterministic performance.

Unfortunately, DFA are difficult to implement effectively as they possess many characteristics which make them spatially complex and memory inefficient [9,10,11,12]. Many attempts have been made to remedy DFA space issues but most are based solely on observations made about DFA test-sets and trends. Thusly, most solutions either present negative performance tradeoffs, or limit the scope of patterns that DFA can be constructed from. There are however, certain DFA characteristics that, when exploited properly, can supersede these limitations.

For this reason, this paper presents a theoretical analysis of the characteristics of DFA that cause complexity and memory issues using two metrics: *State Irregularity* and *State Independence*. State Irregularity for characterizing memory “waste” due to unbalanced *transition* distribution, and State Independence for characterizing memory efficiency due to *state* correlation. With a theoretical understanding of the relationship between these metrics and memory requirements, we present novel DFA implementations that exploit these concepts, culminating in a design that fully exploits all DFA characteristics discussed in this paper, for a minimal DFA implementation that is platform independent.

The remainder of the paper is organized as follows. Deterministic Finite Automata are discussed in Section 2. The concept of State Independence is discussed in Section 3, and State Irregularity in Section 4. Several memory-based DFA implementations that exploit State Independence and State Irregularity are discussed in Section 5. Experimental results for supporting the theory and implementations in this paper are discussed in Section 6. Related work and its relationship to State Independence and State Irregularity is discussed in Section 7 and the paper is concluded in Section 8.

2 Deterministic Finite Automata

2.1 DFA Description

A Deterministic Finite Automaton (D) is defined as follows: $D = \{Q, C, \delta, q_0, F\}$, where:

- Q is a finite set of states
- C is a finite set of input symbols
- δ is a transition function from $Q \times C \rightarrow Q$ for all $q_i \in Q$
- q_0 is the initial state and $q_0 \in Q$
- F is the set of matching states and $F \subseteq Q$

The operation of δ is deterministic and therefore, maximum of one q_j exists for any $\delta(q_i, c) \rightarrow q_j$. This means there is only one outgoing transition per character per state, at the most. Also, $\delta(q_i, c) \rightarrow \{\emptyset\}$ for any $c \notin C$. A result of $\{\emptyset\}$ resets to the initial state q_0 .

Given a string $t \in T$ where T is the set of strings whose symbols compose C , the sequence of symbols in t as input to δ will yield at least one $f_i \in F$ and exactly one $f_i \in F$ where i is unique to t .

2.2 DFA Representation

A DFA can be represented using multiple methods depending on the purpose of that representation. In order to help convey ideas about DFA, we may represent them as a visual graph as seen in Figure 1(a). This figure shows a DFA with $|Q| = 6$. Each directed edge represents $\delta(q_{source}, c_{dest}) \rightarrow q_{dest}$, or a transition from a source state to a destination state on the input character of the destination state. The characters or symbols in C are not shown in this example graph however, it

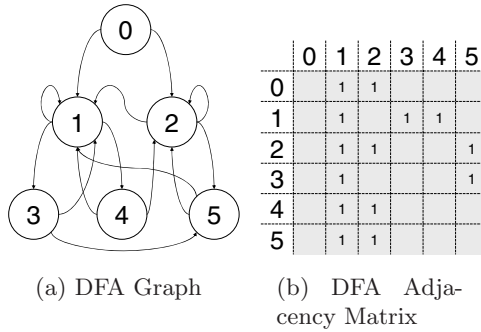


Fig. 1. DFA Representations

can be assumed that a transition to any state is a conditional transition assuming an input character specific to the destination state.

We can also use an adjacency matrix as representation, which is helpful in mathematically extracting properties of DFA. Figure 1(b) shows an adjacency matrix which represents δ of Figure 1(a). Each row represents a source state and each column a destination state. If the intersection of any row and column contains a one, then δ contains a transition $\delta(q_{row}, c_{col}) \rightarrow q_{col}$. If an intersection does not contain a one then $\delta(q_{row}, c_{col}) \rightarrow \{\emptyset\}$.

2.3 DFA Memory

In memory based DFA implementations (including hardware and software), the goal of the transition function δ is to store the appropriate next states for all current states in memory locations retrievable based on information known in δ . We can denote L as the memory space needed to store the results of $\delta(q_i, c)$ for all c for any i . This concept is illustrated by Figure 2. Each state needs L_{q_i} bit memory to store its next state information. Each next state's information is stored in some predefined location in L . So the memory requirements M for any DFA implementation can be generalized by Equation 1, where $|Q|$, is the size of the finite set of states Q . For example, in a standard memory-based DFA implementation where the input character c and the current state q_i are used as the lookup address for the numeric value of the next state, $L = \lceil \log_2(|Q|) \rceil \cdot 2^8$.

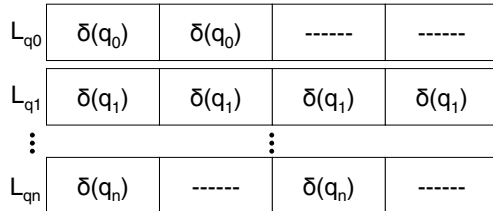


Fig. 2. Abstraction of a memory-based DFA

Next state information must be stored in predefined locations in memory for the sake of lookup and thusly, L in Equation 1, must be equi-sized for all states. This is the nature of the memory in DFA implementations.

$$M = |Q| \cdot L \tag{1}$$

Observing any DFA will show that no matter how a standard DFA is implemented in memory, L will not be regular for all states. In actuality, a summation of L_{q_i} for all $q_i \in Q$, as in Equation 2, will always yield a smaller result than Equation 1, meaning, $M' \leq M$. This means that regularity of M is achieved by growing L_{q_i} for all i to the size of L_{max} which is the maximum L_{q_i} for all i .

$$M' = \sum_{i=1}^{|Q|} L_{q_i} \tag{2}$$

Ideally we want to minimize L_{q_i} for all i and make M and M' equivalent. Two concepts can be used to achieve these goals. State Independence can be exploited in order to reduce L_{q_i} for all i , as independent states can be made to share the same location in L . State Irregularity in δ can be examined and made regular in order to make M and M' naturally equivalent. In the next two sections, we will describe state independence, exploit naturally occurring state independence to minimize memory, describe state irregularity, and eliminate irregularity to further minimize memory.

3 State Independence

Two states q_i and q_j are independent if there exists no $\delta(q_k) \rightarrow q_i$ and $\delta(q_k) \rightarrow q_j$ for any k . In other words, neither q_i nor q_j are adjacent to the head of a directed edge from the same state q_k , where $0 \leq k \leq |Q|$, including i or j . Figure 3(a) shows two states i and j that are independent, whereas Figure 3(b) shows states i and j that are dependent.

State Independence can be used to minimize L_{q_k} for all k . If two states are independent, then δ can store both states in the same location in L as there will never be any $\delta(q_k)$ that results in both q_i and q_j . However, if q_i and q_j are dependent then L_{q_k} must contain q_i and q_j simultaneously, increasing the size of L .

We can estimate the excess percentage of L that is caused by states being dependent using Equation 3. SD gives a general metric for measuring state

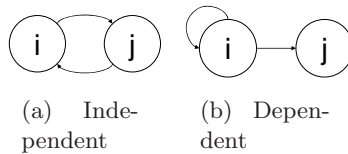


Fig. 3. Example of independent and dependent states

dependence (not independence) for a given DFA. As SD approaches zero, a DFA contains very little dependence and will yield a low width for L , where as a value of close to one for SD , indicates much more dependency and will yield a longer L .

$$SD = 1 - \frac{\lceil \log_2(|Q| - Q_{dep}) \rceil}{Q_{dep} + \lceil \log_2(|Q| - Q_{dep}) \rceil} \tag{3}$$

In Equation 3, Q_{dep} is the number of states in a DFA that are nearly completely dependent. In order to estimate Q_{dep} , we start with adjacency matrix $[A]$ as representation of transition function δ . If we define the fan-in transitions for any state q_i as all $\delta(q_j, c_i) \rightarrow q_i$, where j is all rows in column i of $[A]$ that contain one's, then the fan-in degree sequence (FI_{deg}) for Q can be calculated by Equation 4. In Equation 4, I is identity matrix. Figure 4 shows an example of this calculation. The diagonal of the resulting matrix is the fan-in degree for each state q_{row} . This degree sequence can then be used in Equation 5 to estimate dependent states.

$$FI_{deg} = [A]^T \cdot [A] \cdot I \tag{4}$$

$$Q_{dep} = \frac{\sum_{i=1}^{|Q|} FI_{deg}(i)}{|Q|} \tag{5}$$

3.1 Exploiting State Independence

The first thing necessary for exploiting State Independence is the independence number α of any DFA(D), which is the largest group of mutually independent states in D , can be found by creating a new graph G . Each node in G represents a state in D and if two states in D are independent, their respective nodes in G receive an adjoining, non-directed edge. Given this new graph G , we can find $\alpha(D)$, as $\omega(G) = \alpha(D)$. $\omega(G)$ is the size of the vertex-set of the maximum clique Cl_{max} in G . In summary, this clique represents the largest group of states in D that can be stored in the same location in any L_{q_i} for any i .

We can remove Cl_{max} from G we can again find $\omega(G)$ forming the next largest set of independent states and so on until $G = \{\emptyset\}$. The independent sets in D can

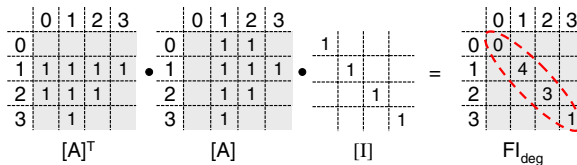


Fig. 4. Calculating Fan-In Degree Sequence with Adjacency Matrix

share the same memory locations in δ helping to minimize the memory needed to implement δ for any D . These independent sets can also be optimized for performance by distributing states such that all states that receive a transition on character c_i are in the same independent set for all i . This optimization is called character aware State Independence.

4 State Irregularity

The quantity of next state information stored in L_{q_i} may be different for each i . This is due to the irregularity of δ for each state. In order to quantify this irregularity, we will again use the fan-in degree sequence from Equation 4.

If we label the largest value in FI_{deg} to be FI_{max} , then we can define State Irregularity(SI_r), by Equation 6. SI_r is the level of irregular edge distribution. As the value of SI_r approaches one, the more irregularity that exists relative to the states with the largest fan-in degrees. This irregularity manifests itself as wasted memory. However, an SI_r value of zero means that each state has the exact same fan-in degree, yielding better memory efficiency. The ideal scenario is if both conditions 1 and 2 from Equation 7 are met.

$$SI_r = \frac{\sum_{i=1}^{|Q|} (1 - \frac{FI_{deg}(i)}{FI_{max}})}{|Q|} \tag{6}$$

$$1.) SI_r = 0 \quad 2.) \sum_{i=1}^{|Q|} FI_{deg}(i) = |Q| \tag{7}$$

So, before exploring solutions to State Irregularity, the causes of such irregularity need be examined. The following are rules for transitions in δ . Following said rules, a discussion of why irregularity is caused by these rules is included. Given:

- DFA, $D = \{Q, C, \delta, q_0, F\}$
- P_x = Set of states contained by all paths in δ that match pattern x .
- K_i = Hierarchy level of each of each $q_i \in Q$ (shortest path from q_0 to each q_i)

Transitions are formed in δ by the following categories:

1. A single transition $\delta(q_i, c_j) \rightarrow q_j$, where $K_j = K_i + 1$, is allowed for each K_j assuming the following:
 - (a) There exists no other $\delta(q_h, c_j) \rightarrow q_j$ where $K_h \leq K_i$
 - (b) If $q_j \in P_x$ then $q_i \in P_x$
2. Failure transitions $\delta(q_i, c_j) \rightarrow q_j$ are allowed assuming the following:
 - (a) $K_j \leq K_i$
 - (b) If $q_j \in P_x$ then $q_i \in P_y$ and $x \neq y$ and $q_i \notin P_x$

3. The following transitions violate transition rules 1 and 2 respectively:

- (a) $\delta(q_i, c_j) \rightarrow q_j$ and $q_i \in P_x$ and $q_j \in P_y$ and $q_i \notin P_y$ and $q_j \notin P_x$ and $K_j > K_i$
- (b) $\delta(q_i, c_j) \rightarrow q_j$ and $\delta(q_i, c_k) \rightarrow q_k$ and $(\delta(q_j, c_k) \rightarrow q_k$ or $\delta(q_k, c_j) \rightarrow q_j)$

Transitions from category 1 are called forward matching transitions. They represent a successful match of the next character in the character sequence of some pattern. These transitions occur from a level K to level $K + 1$ and according to condition 1(a), a state can only receive one forward matching transition. Condition 1(b) states that the state which is adjacent to the tail of the forward matching transition must be in the same set of states for the same pattern as the receiving state.

If patterns have a shared prefix, then those prefixes share states in their pattern sets for all characters in the shared prefix. A branch will occur to two different states after the shared prefix states. Figure 5(a) shows a DFA with forward matching transitions only. The branches seen in this figure occur because of shared prefixes. Branches only occur at states representing the end of a shared prefix and thusly cause State Irregularity.

Transitions from category 2 are called failure transitions. These are transitions from a state on an input character that is not a valid character for the next forward matching transition. Thusly, a transition is followed for said character to a destination state that represents the longest prefix that matches the sequence of characters that caused forward matching up to the state where the failure occurred. As in condition 2(a), these failure transitions must point to a state at the same level or less. Condition 2(b) says that a state cannot have a failure transition which points to a state that is a member of the set of states for the same pattern.

Being that, the longer a prefix is the less likely that a sequence of states will contain that pattern, most failure transitions point to states at level $K = 1$. This creates an irregular distribution of edges, as the greater the value of K for a state, the less likely it is that this state will receive a failure transition. Figure 5(b) shows a DFA with both forward matching transitions and failure transitions.

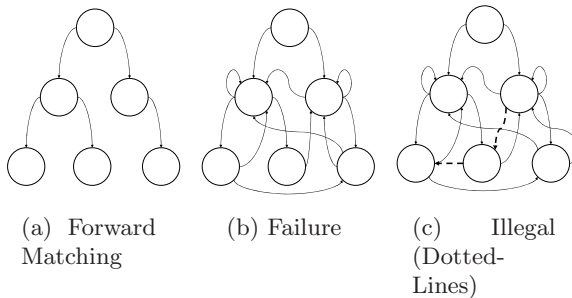


Fig. 5. DFA Transitions

The two types of transitions aforementioned both contribute to State Irregularity. This is partially because some transitions are not allowed. Transitions that conform to conditions 3(a) and 3(b) are illustrated by the dotted edges in Figure 5(c). Condition 3(a) represents a transition that converges from one pattern set to another, which is represented by the forward matching dotted line in Figure 5c. This transition violates conditions 1(a) and 1(b). If this transition were allowed, it would help to balance out irregularity. The same can be said for a transition conforming to condition 3(b), represented by the failure transition dotted line, however, this transition violates condition 2(b). The fact that these transitions violate the structure of DFA makes substantial State Irregularity a typical scenario.

4.1 Normalizing State Irregularity

In order to help correct State Irregularity, DFA can be normalized. This normalization is comparable to database normalization, in which a one-to-one or one-to-many relationship is created between data which belongs to the same set. Figure 6(a) shows a DFA for several patterns. There are a substantial amount of transitions causing irregularity, most of which are failure transitions. Typical in database normalization, is the use of a single lookup record to represent information that would otherwise be repeated many times. The same can be done for transitions in DFA.

For any state that receives more than one transition, a single transition can be kept as a lookup transition for the character needed to transition to that state. The remaining transitions to said state can subsequently be removed. For example, in Figure 6(a), state 1 receives many failure transitions. These transitions always occur on the letter “a”. If the transition $\delta(q_0, a) \rightarrow q_1$ is used as a lookup transition, then a one-to-many relationship is created between all transitions to state 1 and the lookup transition. The same can be done for all transitions $\delta(q_0, c_j) \rightarrow q_j$, yielding the DFA in Figure 6(b). If a state fails for any character, the lookup transitions can be used to determine the appropriate next state.

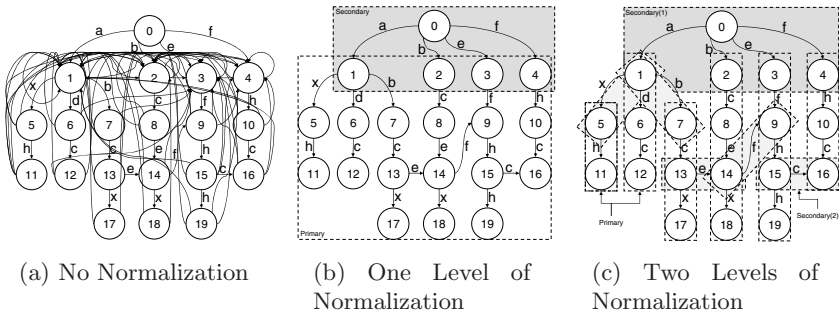


Fig. 6. DFA for Patterns (axh, adc, abcx, bcex, efhh, fhc)

After one level of normalization is applied as in Figure 6(b), the only transitions which still contribute to irregularity are shared prefix branches and multiple character prefix failure transitions. Although typically few in number, these transitions can still contribute a great deal to memory inefficiency, as well as matching operation complexity. Complete normalization can be achieved by adding another level of normalization. After completing this normalization as seen in Figure 6(c), each lookup block is completely regular relative to itself. When storing δ in memory for a normalized DFA, there is little to no wasted memory due to State Irregularity.

5 Implementations

This section presents two memory-based DFA implementations that exploit the characteristics and optimizations discussed in previous sections. Both implementations in Sections 5.1 and 5.2 are solely memory-based in that they only require memory, no special hardware is required. Both implementations also exploit State Independence to minimize memory. The first implementation normalizes DFA to one level of normalization. The second in Section 5.2, uses two levels of normalization and has a very simple lookup structure. It is the culmination of all concepts discussed in this paper. It is very memory efficient and executes quickly.

5.1 Portable Predictive Pattern Matching Finite State Machine

Portable Predictive Pattern Matching Finite State Machine (P^3FSM) is ideal for implementation in software. It utilizes one level of DFA normalization. For this example, we will refer to the DFA in Figure 6(b). From this DFA, State Independence is determined. This determines what location(loc) the each next state information will be stored in as in Table 1, which shows the primary memory contents. Table 2 shows the secondary memory contents. Each location(loc) in Table 1 consists of a character value in the left column and a next state value in the right column. In order to keep L_{q_i} small for all i , the character and next state values are generated relative to the location(loc) which means that values can be reused, making them shorter.

The operation of P^3FSM consists of a short series of simple lookups as illustrated by the following examples:

Example 1. Primary Memory: Assume the current state is q_3 and the next input character is “f”, or $\delta(q_3, f)$. We first lookup the information for “f” in Table 2. With this information we move to Table 1 at adrs(8) which is the address of L_{q_3} . At adrs 8, we now compare the val for “f” which is 100, to the char val in loc(1) which is also 100. Since both values match, we extract the state val at adrs(8)/loc(1) which is 010. We add this state val to the offset for “f” which is 9, yielding: $9 + 010 = eleven$. This means *eleven* is the address of the next state. At adrs(11) in Table 1 lies L_{q_9} . Now, if we reference the DFA from Figure 6(b), we see the transition $\delta(q_3, f) \rightarrow q_9$.

Example 2. Secondary Memory: If we redo the same scenario except using a different input character such as $\delta(q_3, b)$, the comparison of the val for “b”

Table 1. Primary Memory Content (Q long by L wide)

	Adrs	Loc(1)	Loc(2)	Loc(3)
L_{q_1}	1	001	010	1 001 0 1
L_{q_2}	2	010	001	0 000 0 0
L_{q_7}	3	010	011	0 000 0 0
L_{q_8}	4	011	010	0 000 0 0
$L_{q_{12}}$	5	000	000	0 000 0 0
$L_{q_{13}}$	6	011	010	1 010 0 0
$L_{q_{16}}$	7	000	000	0 000 0 0
L_{q_3}	8	100	010	0 000 0 0
$L_{q_{14}}$	9	100	010	1 011 0 0
L_{q_4}	10	000	000	0 001 0 0
L_{q_9}	11	000	000	0 011 0 0
$L_{q_{10}}$	12	010	100	0 000 0 0
$L_{q_{11}}$	13	000	000	0 000 0 0
$L_{q_{15}}$	14	000	000	0 100 0 0
$L_{q_{19}}$	15	000	000	0 000 0 0
L_{q_5}	16	000	000	0 010 0 0
$L_{q_{17}}$	17	000	000	0 000 0 0
$L_{q_{18}}$	18	000	000	0 000 0 0
L_{q_6}	19	010	010	0 000 0 0

Table 2. Secondary Memory Content

Char	Val	Loc	Offset	Adrs
a	000	1	0	1
b	001	1	1	2
c	010	1	3	0
d	0	3	18	0
e	011	1	7	8
f	100	1	9	10
h	0	2	11	0
x	1	2	15	0

would not match the char val at $\text{adrs}(8)/\text{loc}(1)$. Therefore, we chose as the next state, the adrs for b from Table 2, which is 2. This means the system chooses $\delta(q_3, b) \rightarrow q_2$. This transition is not present in Figure 6(b), but does not need to be due to the normalization process.

5.2 Simple Instruction Finite State Machine

Simple Instruction Finite State Machine (SI-FSM) is also ideal for implementation in software. It utilizes two levels of DFA normalization. For this example, we will refer to the DFA in Figure 6(c). From this DFA, the primary memory content is populated by storing L_{q_i} for each i in the order they appear in their respective primary blocks in Figure 6(c). The primary memory content is shown in Table 3. The secondary(1) memory contents are shown in Table 4. Since this implementation uses two levels of normalization, an extra secondary block is used as shown in Table 5. This table is populated much the way the primary table is in P^3FSM .

The operation of SI-FSM consists of a short series of simple lookups as illustrated by the following examples:

Example 1. Primary Memory: Assume the current state is q_3 and the next input character is “f”, or $\delta(q_3, f)$. We can immediately go to Table 3 at $\text{adrs}(13)$ and compare the input character “f” to the char at $\text{adrs}(13)$, which is also “f”. Since there is a match, the adrs is simple incremented moving the current state

Table 3. Primary Memory Content

	Adrs	Char	Fail
L_{q_5}	1	h	0
$L_{q_{11}}$	2	—	0
L_{q_1}	3	d	1
L_{q_6}	4	c	0
$L_{q_{12}}$	5	—	0
L_{q_7}	6	c	0
$L_{q_{13}}$	7	x	2
$L_{q_{17}}$	8	—	0
L_{q_2}	9	c	0
L_{q_8}	10	e	0
$L_{q_{14}}$	11	x	3
$L_{q_{18}}$	12	—	0
L_{q_3}	13	f	0
L_{q_9}	14	h	0
L_{q_5}	15	h	4
$L_{q_{19}}$	16	—	0
L_{q_4}	17	h	0
$L_{q_{10}}$	18	c	0
$L_{q_{16}}$	19	—	0

Table 4. Secondary(1) Memory Content

Char	Loc	Adrs
a	0	3
b	1	9
c	1	0
d	0	0
e	1	13
f	1	17
h	0	0
x	2	0

Table 5. Secondary(2) Memory Content

Fail	Loc(1)	Loc(2)
1	b 6	x 1
2	e 11	— —
3	f 14	— —
4	c 19	— —

to adrs(14) which is L_{q_9} . We know from before that $\delta(q_3, f) \rightarrow q_9$ is the correct transition.

Example 2. Secondary(1) Memory: If we redo the same scenario except using a different input character such as $\delta(q_3, b)$, the comparison of “b” would not match the char “f” adrs(13). Therefore, the 0 in the fail field at adrs(13) tells us to use the adrs for “b” from Table 4 which is 9. Adrs(9) is the address for q_2 which yields the correct transition $\delta(q_3, b) \rightarrow q_2$.

Example 3. Secondary(2) Memory: Assume input $\delta(q_1, b)$. We compare the incoming character “b” to the character at adrs(3) in Table 3. The characters do not match $b \neq d$, so we use the value of 1 from the fail field and move to the fail index of 1 in Table 5. We compare the incoming character “b” to the “b” in loc(1) and find a match. Are next state address becomes adrs(6) which is L_{q_7} . This means a transition $\delta(q_1, b) \rightarrow q_7$ has occurred which is the correct transition.

6 Results

DFA contain varying levels of State Independence and State Irregularity in a natural form, that is, with no optimizations. These characteristics drastically affect memory requirements for different DFA. In Section 6.1 we begin by examining naturally occurring State Independence and State Irregularity, and their relationship to each other and to memory. We proceed in Section 6.2 to show

how normalization works to correct State Irregularity and enhance State Independence, thusly decreasing memory requirements.

6.1 Non-normalized DFA

To start out we will examine the correlation between *SD* and *SR*, as well as *SD* and *SR* versus DFA size. Figure 7 shows the values for *SD* and *SR* given DFA constructed from 1 to 500 patterns. The patterns used to build the DFA are randomly extracted from Snort [13] rules. Several observations can be made about *SD* and *SR* from this figure.

Although State Independence occurs naturally in DFA, improving State Irregularity helps improve State Independence. In this figure *SD* and *SR* are moderately parallel. As State Irregularity Increases, so does the dependence between states. The transitions that promote *SR* such as shared prefix branches and failure transitions do contribute to dependency between states. That is why an increase in *SD* is observed as *SR* increases as well.

Another observation that can be made from Figure 7 is based on the fact that, as the number of patterns, or size of the DFA, increases, the number of failure transitions to single character prefixes increase. This both decreases the transition balance and increases dependency between states. So as *SR* increases, *SD* increases.

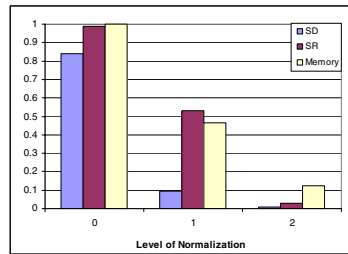
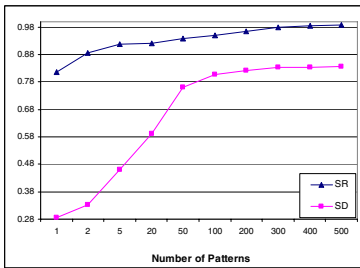


Fig. 7. State Dependence(SD) and **Fig. 8.** Effects of Normalization (mem-State Irregularity(SR) vs. DFA Size in ory relative to zero normalization) Patterns

Table 6. Group of snort ruleset characteristics (memory data relative to synthetic 1)

Ruleset	Patterns	Characters	States	SD	SR	Memory-per-Char	Memory-per-Pattern
Synthetic 1	1	20	20	0.286	0.815	1.0	1.0
Synthetic 2	2	37	37	0.333	0.885	1.4	1.3
Synthetic 3	5	111	93	0.462	0.918	2.9	3.2
Synthetic 4	20	363	302	0.591	0.921	5.4	4.9
IMAP	33	232	181	0.619	0.936	5.4	4.8
FTP	58	343	296	0.700	0.942	8.6	4.5
Policy	88	1009	875	0.767	0.963	10.4	6.0
Web-CGI	200	2361	1733	0.773	0.989	10.5	6.2

Next we will examine how SD and SR effect memory. To illustrate this, refer to Table 6 which shows characteristics for different Snort rulesets. For each ruleset, the table shows the number of patterns and characters in those patterns, as well as the number of states in the resulting DFA. SD and SR are calculated for each ruleset. Lastly, the memory-per-character and memory-per-pattern fields show the amount of memory required relative to the memory results for the Synthetic 1 ruleset. For example, IMAP requires 5.4 times the memory per character than Synthetic 1. Memory per character and per pattern are used because each ruleset has a different number of patterns and characters. The results are relative to Synthetic 1 so the effects of SD and SR are more easily observed.

In Table 6, similar trends can be seen for SD and SR as in Figure 7. This table reveals how these values effect memory. As SD increases, more memory (relative to DFA size) is required to implement patterns in a DFA. The same can be said for SR , as more memory is needed as SR increases. A small increase in SR means an increase in the percentage of states that are dependent, hence the increase in SD . This increase in SD yields a longer L , or memory width, due to the dependent states needed to be stored in separate locations in L . The end result is empty or wasted memory for many states as shown in Figure 2. Adjusting SR through normalization as in Section 4.1 allows for nearly complete reduction of SR and SD for a minimal L , or memory width. This is demonstrated in the next section.

The difference in memory is also substantial per pattern, however, many factors can effect this such as; the number of shared prefixes in the patterns, similarity in non-prefix substrings in patterns and length of patterns. Therefore, memory per character is a better metric for observing the effect of State Independence and State Irregularity.

6.2 Normalized DFA

State Independence can be utilized to reduce the memory for DFA implementations. Also, State Irregularity can be reduced, while State Independence increased through normalization as discussed in section 4.1. Figure 8 shows SD , SR and $memory$ for zero, one and two levels of normalization. The data comes from a 500 pattern DFA derived from Snort rulesets. The value of the memory is relative to memory for zero levels of normalization.

Without normalization, SD is around 0.84. After one level of normalization SD decreases to 0.09 and when the DFA is completely normalized at two levels, SD reduces to less than 0.01, which essentially means almost complete independence between states. This is because normalization reduces dependency between states as well as irregularity in transition distribution.

The most important observation to be made from Figure 8, is that the memory is reduced to about 47 percent of the original requirement after one level of normalization and to about 12 percent after two levels of normalization. These values for memory are especially exciting because the base value for memory with zero levels of normalization already takes state independence into account.

So the decrease to 47 percent(1 level) and then 12 percent(2 levels) comes solely from the increase in State Independence due to normalization.

7 Related Work

Soewito and etc. [14] introduced a memory-based hardware DFA implementation called SAM-FSM, or Self-Addressable Memory-based Finite State Machine. SAM-FSM is the first approach to memory-based DFA to explicitly exploit State Independence. It groups states into independent sets. Those sets can share a location in the so called self-addressable codes stored in memory for each state. These codes are synonymous to L_{q_i} in this paper.

Kumar and etc. [10] introduced a method for DFA minimization called D^2FA . This method points many redundant failure transitions toward a default state. It allows many failure transitions that fan-out from a single state to many other states, to become one transition to a default state. In D^2FA however, many default states are allowed. Since a transition to a default state means not advancing to the next input character, but instead holding the character for processing at the default state, if multiple default transitions are followed in a row, it may be many clock cycles before a character is processed and the input character is advanced. To circumvent this processing cost, Becchi and Crowley [9] introduces an optimization which places a bound on the number of default states that can be followed in a row. This bound slightly limits the transition reduction capability, however State Independence is not exploited by D^2FA .

Lunteren [11] introduced a string-matching engine, BFPM which uses a prioritized list of state transition rules stored in memory. These prioritized rules are very similar to our normalization techniques. Occasionally, lookup needs to take place between the clustered sections. BFPM does not exploit State Independence. Also, the normalization used is not complete and there may be multiple transitions spanning clustered sections.

8 Conclusion

This paper presents a thorough analysis of DFA in order to understand the relationship between DFA characteristics and memory requirements. Two concepts related to minimization of memory-based DFA are introduced: State Independence and State Irregularity. State Independence allows next state information for DFA to be stored in the same location in memory, minimizing memory width. State Irregularity is a characteristic of DFA that can be normalized to reduce memory waste. We present experimental results using realistic pattern sets, that demonstrate the affect these characteristics have on memory requirements.

With a clear understanding of the relationship between DFA characteristics and memory, we show how to optimize DFA to exploit these characteristics for enhanced memory reductions. We believe that our work bridges many gaps between existing heuristic minimization methods, in order to help solve the overall problem of memory-based, deterministic pattern matching for scalable Network Intrusion Detection Systems.

References

1. Aho, A., Corasick, M.: Efficient string matching: An aid to bibliographic search. *Communications of the ACM* 18 (1975)
2. Brodie, B.C., Taylor, D.E., Cytron, R.K.: A scalable architecture for high-throughput regular-expression pattern matching. *SIGARCH Comput. Archit. News* 34(2), 191–202 (2006)
3. Roesch, M.: Snort – lightweight intrusion detection for networks. In: *Proc. of the 13th Systems Administration Conference* (1999)
4. Varghese, G.: *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices*, 1st edn. Morgan Kaufmann, San Francisco (2005)
5. Bu, L., Chandy, J.A.: Fpga based network intrusion detection using content addressable memories. In: *FCCM 2004: Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, Washington, DC, USA, pp. 316–317 (2004)
6. Tuck, N., Sherwood, T., Calder, B., Varghese, G.: Deterministic memory-efficient string matching algorithms for intrusion detection. In: *Proc. of the IEEE Infocom Conference*, pp. 333–340 (2004)
7. Yu, F., Chen, Z., Diao, Y., Lakshman, T., Katz, R.H.: Fast and memory-efficient regular expression matching for deep packet inspection. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-76* (May 2006)
8. Fisk, M., Varghese, G.: Applying fast string matching to intrusion detection, los Alamos National Lab Report (2002)
9. Becchi, M., Crowley, P.: An improved algorithm to accelerate regular expression evaluation. In: *Proc. of ANCS*, pp. 145–154. ACM, Orlando (2007)
10. Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., Turner, J.: Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In: *Proc. of ACM SIGCOMM*, pp. 339–350. ACM, New York (2006)
11. van Lunteren, J.: High-performance pattern-matching for intrusion detection. In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–13 (2006)
12. Smith, R., Estan, C., Jha, S., Kong, S.: Deflating the big bang: fast and scalable deep packet inspection with extended finite automata. In: *SIGCOMM 2008: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, New York, USA, pp. 207–218 (2008)
13. Snort Rule Database, Snort (2009), <http://www.snort.org/pub-bin/downloads.cgi>
14. Soewito, B., et al.: Self addressable memory-based fsm (sam-fsm): A scalable intrusion detection engine. *IEEE Network Special Issue on Recent Developments in Network Intrusion Detection* 23(1), 14–21 (2009)

A LoSS Based On-line Detection of Abnormal Traffic Using Dynamic Detection Threshold

Zhengmin Xia¹, Songnian Lu^{1,2}, Jianhua Li^{1,2}, and Aixin Zhang²

¹ Department of Electronic Engineering,
Key Lab of Information Security Integrated Management Research,
Shanghai Jiao Tong University, 200240,
Shanghai, P.R. China

² School of Information Security Engineering,
Key Lab of Information Security Integrated Management Research,
Shanghai Jiao Tong University, 200240, Shanghai, P.R. China
{miaomiaoxzm, snlu, lijh888, axzhang}@sjtu.edu.cn

Abstract. Abnormal traffic detection is a difficult problem in network management and network security. This paper proposed an abnormal traffic detection method based on LoSS (loss of self-similarity) through comparing the difference of Hurst parameter distribution under the network normal and abnormal traffic time series conditions. This method adopted wavelet analysis to estimate the Hurst parameter of network traffic in large time-scale, and the detection threshold could self-adjusted according to the extent of network traffic self-similarity under normal conditions. The test results on data set from Lincoln Lab of MIT demonstrate that the new detection method has the characteristics of dynamic self-adaptive and higher detection rate, and the detection speed is also improved by one time segment.

Keywords: Network traffic, Anomaly detection, Hurst parameter, Discrete time series, Self-similarity.

1 Introduction

Along with the development and popularization of network, more and more serious attacks have been present and threat to the stability of the internet. It can be seen that network traffic related to attacks, especially distributed denial-of-service (DDoS) attacks, is “pulse” from the perspective of dynamical aspects for limited time interval in physics[1], so we call these attack-contained traffic abnormal traffic and attack-free traffic normal traffic. The abnormal traffic can cause a huge harm to the network performance, mainly reflect in two aspects: (i) consumes a large amount of network bandwidth resource, and causes network congestion, therefore results in network packet loss rate increases, and prolongs the transmission delay-time; (ii) occupies network equipment system resources (CPU, memory etc.), therefore leads the network unable to provide normal services. So it is necessary to detect and prevent this abnormal traffic timely and successfully.

In 1993, Leland et al.[2] first found the local network traffic had the nature of self-similarity and long-range dependence (LRD). Paxson et al.^[3] also found the wide area network traffic had the self-similarity nature. Self-similar is the property that associated with the objects whose structure is unchanged at different time scales. The work presented in [4] showed that the self-similarity of internet traffic distributions could often be accounted for a mixture of the actions of a number of individual users, hardware and software behaviors at their originating hosts, multiplexed through an interconnection network.

Li[5] quantitatively described the abnormal traffic statistics, and found that the averaged Hurst parameter of abnormal traffic usually tended to be significantly smaller than that of normal traffic. The works in [6][7][8][9] presented new methods of detecting the possible presence of abnormal traffic without a template of normal traffic. These methods used LoSS definition with the self-similarity or Hurst parameter beyond the fixed threshold, but the real-world network traffic self-similarity extent is changing over time, so these detection methods had low detection rate and high false alarm rate. In this paper, we present a dynamic self-adaptive abnormal traffic detection method based on the self-similar nature, and the detection threshold can change with the network traffic self-similarity extent automatically. The method mainly includes three steps: (i) estimates the Hurst parameter of the incoming network traffic using wavelet analysis, and (ii) decides whether the incoming traffic is normal or not by comparing the Hurst parameter with the detection threshold, then (iii) updates the detection threshold if the incoming traffic is normal, otherwise informs the network administrators. This method has the characteristics of dynamic self-adaptive and convenience to be implemented.

The remainder of this paper is organized as follows. Section 2 briefly introduces the theoretical background of self-similarity and wavelet-based Hurst parameter estimation. Section 3 describes the detection principle and explains the abnormal traffic detection process in detail. The detection results of the data set from Lincoln Lab of MIT are presented in Section 4. Finally, a brief summary of our work and future research are provided in section 5.

2 Self-similarity and Hurst Parameter Estimation

2.1 A Brief Review of Self-similarity

Self-similarity means that the sample paths of the process $X(t)$ and those of rescaled version $c^H X(t/c)$, obtained by simultaneously dilating the time axis by a factor $c > 0$, and the amplitude axis by a factor c^H , cannot be statistically distinguished from each other. H is called the self-similarity or Hurst parameter. Equivalently, it implies that an affine dilated subset of one sample path cannot be distinguished from its whole.

Let $X = \{X_i, i \in \mathbb{Z}_+\}$ be a wide-sense stationary discrete stochastic process with constant mean μ , finite variance σ^2 , and autocorrelation function $r(k), (k \in \mathbb{Z}_+)$. Let $X^{(m)}$ be a m-order aggregate process of X ,

$$X_i^{(m)} = (X_{mi-m+1} + \dots + X_{mi})/m \quad i, m \in \mathbb{Z}_+ . \tag{1}$$

For each m , $X^{(m)}$ defines a wide-sense stationary stochastic process with autocorrelation function $r^{(m)}(k), (k \in \mathbb{Z}_+)$.

Definition 1. A second-order stationary process X is called exactly second-order self-similar (ESOSS) with Hurst parameter $H=1-\beta/2$, $0 < \beta < 1$, if the autocorrelation function satisfies

$$r^{(m)}(k) = r(k), \quad (k, m \in \mathbb{Z}_+) . \tag{2}$$

Definition 2. A second-order stationary process X is called asymptotical second-order self-similar (ASOSS) with Hurst parameter $H=1-\beta/2$, $0 < \beta < 1$, if the autocorrelation function satisfies

$$\lim_{m \rightarrow \infty} r^{(m)}(k) = r(k), \quad (k \in \mathbb{Z}_+) . \tag{3}$$

where $r(k) = [(k+1)^{2-\beta} - 2k^{2-\beta} + (k-1)^{2-\beta}]/2$. In the field of network traffic theory, it is more practical to use ASOSS.

2.2 On-line Hurst Parameter Estimation

Several methods had been developed to estimate the Hurst parameter, such as aggregated variance^[10], local whittle^[11], and the wavelet-based methods^[12]. By far, the wavelet-based estimator of the Hurst parameter stands out as one of the most reliable estimators in practice for it is more robust with respect to smooth polynomial trends and noise^[13]. Wavelet-based Hurst parameter estimation mainly includes three methods: wavelet variance analysis, wavelet power spectra analysis, and wavelet energy analysis. These methods are consistent in essence. In this section, the wavelet energy analysis method is briefly introduced, and more details please refer to [12].

For a given traffic trace time series X , Hurst parameter H can be estimated as follows:

- First, for each scale j and position k , compute the wavelet coefficients:

$$d(j, k) = \langle X, \Psi_{j,k}(n) \rangle = \sum_{n=1}^{\infty} X \Psi_{j,k}(n) . \tag{4}$$

where $\Psi_{j,k}(n) = 2^{-j/2} \Psi_0(2^{-j}n - k)$ and Ψ_0 is the (Daubechies) mother wavelet.

- Second, compute the wavelet energy μ_j for each scale j :

$$\mu_j = \frac{1}{N_j} \sum_{k=1}^{N_j} d^2(j, k) . \tag{5}$$

where N_j is the total number of wavelet coefficients at scale j .

- Then, make a plot of $\log_2(\mu_j)$ versus scale j and apply linear regression over the curve region that looks linear, and compute the slope α .
- Finally, estimate the Hurst parameter as:

$$H \hat{=} (\alpha + 1)/2 . \tag{6}$$

3 Abnormal Traffic Detection

3.1 Detection Principle

For given discrete time series $X = \{X_i, i \in \mathbb{Z}_+\}$, $Y = \{Y_i, i \in \mathbb{Z}_+\}$ and $Z = \{Z_i, i \in \mathbb{Z}_+\}$, let X and Y be normal traffic and abnormal traffic respectively, and Z be the attack traffic during transition process of attacking. X and Z are uncorrelated^[5], so Y can be abstractly expressed by $Y = X + Z$.

Fig. 1 illustrates the components of normal and abnormal traffic, $x_k(i)$ represents the number of bytes send out by node k at time i for normal network services, and $z_q(i)$ represents the number of bytes send out by node q at time i for network attacks, and Y_i be total traffic the target received at time i .

Let r_{XX} , r_{ZZ} and r_{YY} be autocorrelations of X , Z and Y respectively. During the transition process of attacking, $\|r_{YY} - r_{XX}\|$ is noteworthy^[5], and $r_{YY} = r_{XX} + r_{ZZ}$. For each value of $H \in (0.5, 1]$, there is exactly one autocorrelation function with self-similarity as it can be seen from Beran (1994, p. 55). Thus, a consequence is that $\|H_Y - H_X\|$ is considerable, where H_Y and H_X are average Hurst parameters of Y and X , respectively. Hence, H is a parameter that can yet be used to describe the abnormality of network traffic.

3.2 Detection Process

In general, the network traffic of one host or one LAN is almost equal and remains steady as a whole under normal conditions for a period of time. But for longer time, the traffic is unstable, and it changes with the network circumstance (network load, and number of users, et al.), so the detection threshold also should change accordingly. Figure 2 is the flowchart of self-adaptive abnormal traffic detection.

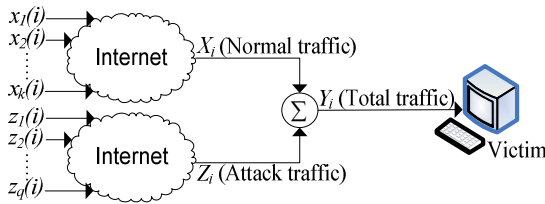


Fig. 1. Composition of normal and abnormal traffic

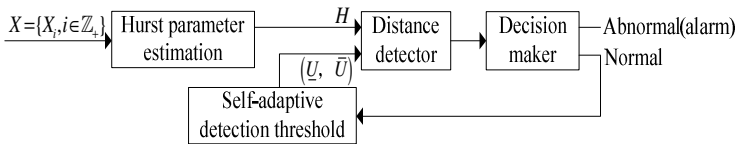


Fig. 2. Self-adaptive abnormal traffic detection process

The detailed detection process is as follows:

Step one: detection threshold initialization

Before the detection, the detection threshold should be initialized. According to [8], the typical Hurst parameter of network traffic is 0.75, and the traffic will be considered as abnormal if the change of Hurst parameter is more than 0.15. So, let the down threshold \underline{U} and up threshold \bar{U} be 0.6 and 0.9, respectively.

Step two: abnormal traffic detection

Divide the traffic X received at time segment N into M non-overlapping blocks. Each block is a series of L length. Let $H_N(m)$ ($m=1,2,\dots, M$) be the Hurst parameter of each block. Averaging $H_N(m)$ in terms of index m yields

$$H_N = \frac{1}{M} \sum_{m=1}^M H_N(m) . \tag{7}$$

H_N represents the Hurst parameter of time segment N . The traffic X is considered normal if $H_N \in (\underline{U}, \bar{U})$ and goes to step three, otherwise X is considered abnormal, and then goes to step one, and reports the abnormal to the network administrators at the same time.

Step three: detection threshold updating

When the traffic X is considered normal, the detection threshold should be updated before the next detection. Considering the Hurst parameter H_n ($n=1,2,\dots, N$) of time segment N and $N-1$ consecutive time segments before time segment N , a normality assumption for H_n is quite accurate in most cases for $M \geq 10$ (Bendat and Piersol, 1986) .

Let \bar{H} and σ_H^2 be the mean and variance estimates of H_n separately,

$$\begin{aligned} \bar{H} &= E[H_n] = \frac{1}{N} \sum_{n=1}^N H_n \\ \sigma_H^2 &= \frac{1}{N} \sum_{n=1}^N H_n^2 - \bar{H}^2 \end{aligned}$$

Then, the probability density function of H_n can be expressed as

$$f(H_n; \bar{H}, \sigma_H^2) = \frac{1}{\sqrt{2\pi\sigma_H}} e^{-\frac{[H_n - \bar{H}]^2}{2\sigma_H^2}} . \tag{8}$$

The confidence interval of H_n with $(1-\alpha)$ confidence coefficient is given by $(\bar{H} - \sigma_H z_{\alpha/2}, \bar{H} + \sigma_H z_{\alpha/2})$. Considering the self-similar network traffic's Hurst parameter is located between $(0.5, 1]$, so the down threshold \underline{U} and up threshold \bar{U} could be set as:

$$\begin{aligned} \underline{U} &= \max(0.5, \bar{H} - \sigma_H z_{\alpha/2}) \\ \bar{U} &= \min(\bar{H} + \sigma_H z_{\alpha/2}, 1) \end{aligned} . \tag{9}$$

3.3 Detection Performance

One performance of the proposed method is that, you can control the false alarm probability (p_f) as low as you wish. The term false alarm means mistakenly recognizing a normal traffic as abnormal traffic.

In step three of section 3.2, we assumed that H_n satisfy a normal distribution in normal condition, so the false alarm probability p_f can be expressed by

$$\begin{aligned}
 p_f &= p(H_n < \underline{U}) + p(\bar{U} < H_n) \\
 &= \int_{-\infty}^{\underline{U}} \frac{1}{\sqrt{2\pi}\sigma_H} e^{-\frac{[H_n - \bar{H}]^2}{2\sigma_H^2}} dH_n + \int_{\bar{U}}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma_H} e^{-\frac{[H_n - \bar{H}]^2}{2\sigma_H^2}} dH_n \\
 &= \alpha
 \end{aligned}$$

This means, you can control the false alarm probability by setting appropriate value of α .

4 Experiment

4.1 Data Preparation

To testing the proposed method, we used the traffic data set from Lincoln Lab of MIT, named DARPA 2000 LL_DDoS_2.0.2. It collected from U.S. Air Force base over a span of approximately 1 hour 45 minutes on April 16 2000. This data set includes five phases of DDoS attack: probe a host; break in-to the host; upload DDoS software and attack script; initiate attack; launch the DDoS. Some of the traffic is displayed in figure 3, and the merge time scale is 100ms.

4.2 Experimental Results and Analysis

Let $M=N=10$, and $L=128$, so the total data set can be divided into 48 time segments, and each time segment lasts about 128s, that means every segment has 10 blocks, and each block contains 128 datas. We use Daubechies(3)[8] as mother

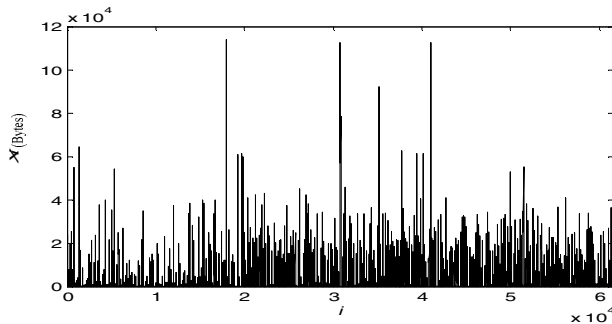


Fig. 3. Traffic of LL_DDoS_2.0.2

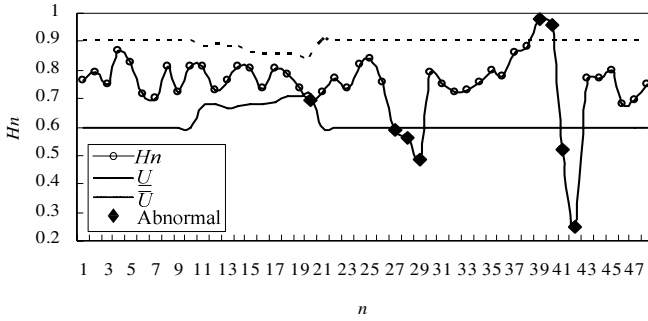


Fig. 4. Change trend of H_n , \underline{U} and \bar{U} ($L=128, \alpha=0.05$)

wavelet to estimate the Hurst parameter of each block. The change trend of H_n , \underline{U} and \bar{U} are displayed in figure 4.

In figure 4, time segments 1~10 are the initialization of the detection threshold, so the down threshold \underline{U} and up threshold \bar{U} could be set as 0.6 and 0.9, respectively. At time segment 11, the traffic of 10 consecutive time segments before time segment 11 is normal, so the updated \underline{U} and \bar{U} could be set as 0.67477 and 0.87885 respectively with 95% confidence level ($\alpha=0.05$), and $H_{11} \in (\underline{U}, \bar{U})$, so the traffic of time segment 11 is considered normal. At time segment 20, the updated \underline{U} and \bar{U} could be set as 0.71306 and 0.884606 respectively with the same confidence level, and $H_{20} \notin (\underline{U}, \bar{U})$, so the traffic of time segment 20 is considered abnormal. Report this abnormal to network administrators, and reinitialize the detection threshold of 10 consecutive time segments after the time segment 20. Using this detection method, the traffic is considered abnormal at time segments 20, 27-29, and 39-42.

Change the confidence level to 99% ($\alpha=0.01$), so the H_n , down threshold \underline{U} and up threshold \bar{U} of each time segment are displayed in figure 5. From figure 5, we can see that the traffic is considered abnormal at time segments 27-29, and 39-42.

Comparing the figure 4 and figure 5, we can find that the wider the confidence level is, the less abnormal traffic will be detected; the narrower the confidence level

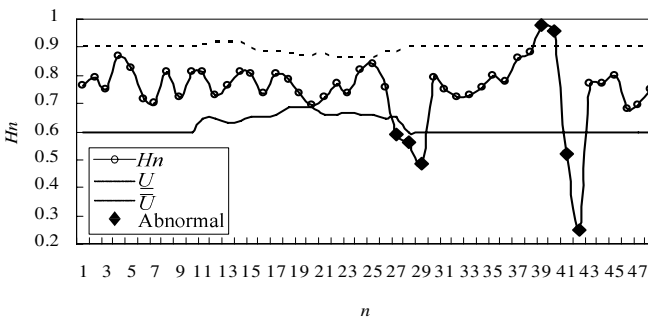


Fig. 5. Change trend of H_n , \underline{U} and \bar{U} ($L=128, \alpha=0.01$)

is, the more abnormal traffic will be detected. This is because under the condition of wider confidence level, some low-rate abnormal traffic will be missed, and under the condition of narrower confidence level, some normal change of traffic will be taken as abnormal.

Let $L=256$, so the total data set can be divided into 24 time segments, and each time segment lasts about 256s. We still use Daubechies(3) as mother wavelet to estimate the Hurst parameter of each block. The change trend of H_n , \underline{U} and \bar{U} are displayed in figure 6.

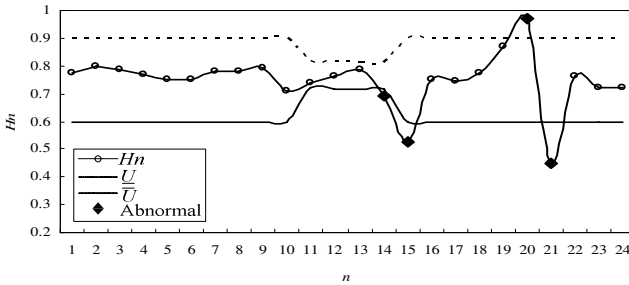


Fig. 6. Change trend of H_n , \underline{U} and \bar{U} ($L=256, \alpha=0.05$)

By detection, the traffic is considered abnormal at time segments 14-15, and 20-21, corresponding to time segments 28-30, and 40-42 when $L=128$.

Comparing the figure 4 and figure 6, we can find that the longer the L is, the less abnormal traffic will be detected; the shorter the L is, the more abnormal traffic will be detected. This is because when L is long, some short-time abnormal traffic will be missed, and when L is short, the data used to estimate the Hurst parameter is less, so the accuracy of Hurst parameter estimation will be degrade and lead to the false detection.

According to Ref.[8], we use Daubechies(3) to estimate Hurst parameter of every time segment directly when $L=256$, and use the fix down threshold $\underline{U} = 0.6$ and up threshold $\bar{U} = 0.9$ to detect the abnormal traffic, the results is displayed in figure 7.

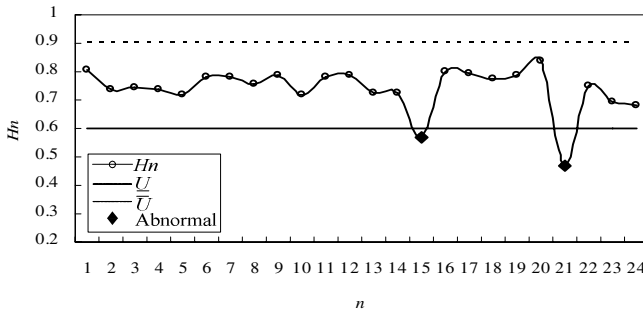


Fig. 7. Change trend of H_n , \underline{U} and \bar{U} ($L=256$)

Through detection, the traffic is considered abnormal at time segments 15 and 21. Comparing figure 6 and figure 7, we can find that the detection speed of self-adaptive threshold detection method is faster than fixed threshold detection method one time segment (256s). This will bring a great benefit to network administrators for early finding out abnormal traffic and taking effective measures to prevent more damages to the network.

5 Conclusion

This paper compared the difference of Hurst parameter distribution under network normal and abnormal traffic time series conditions, and designed an abnormal traffic detection method using self-adaptive detection threshold. The detection results on data set DARPA 2000 LL_DDoS_2.0.2 demonstrated that the new detection method had the characters of self-adaptive and higher detection rate, and the detection speed is also improved. These merits will bring a great benefit to network administrators for early find out abnormal traffic and take effective measures to prevent more damages to the network. Recent research found that wireless network traffic also had the character of self-similarity, so our future research will focus on abnormal traffic detection of wireless network.

References

- [1] Toma, G.: Practical Test Functions Generated by Computer Algorithms. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3482, pp. 576–584. Springer, Heidelberg (2005)
- [2] Leland, W.E., Taqqu, M.S., Willinger, W., et al.: On the Self-similar Nature of Ethernet Traffic (extended version). *IEEE/ACM Transactions on Networking* 2(1), 1–15 (1994)
- [3] Paxson, V., Floyd, S.: Wide area traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking* 3(3), 226–244 (1995)
- [4] Mark, E., Azer, B.: Self-similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking* 5(6), 835–846 (1997)
- [5] Li, M.: Change Trend of Aaveraged Hurst Parameter of Traffic Under DDOS Flood Attacks. *Computers & security* 25(3), 213–220 (2006)
- [6] Schleifer, W., Mannle, M.: Online Error Detection Through Observation of Traffic Self-similarity. *IEE Proceedings on Communications* 148(1), 38–42 (2001)
- [7] William, H., Gerald, A.: The Loss Technique for Detecting New Denial of Service Attacks. In: *IEEE Proceedings on SoutheastCon*, pp. 302–309. IEEE Press, Los Alamitos (2004)
- [8] Ren, X., Wang, R., Wang, H.: Wavelet Analysis Method for Detection of DDOS Attack on the Basis of Self-similarity. *Frontiers of Electrical and Electronic Engineering in China* 2(1), 73–77 (2007)
- [9] Mohd, F., Mohd, A., Ali, S., et al.: Uncovering Anomaly Traffic Based on Loss of Self-similarity Behavior Using Second Order Statistical Model. *International Journal of Computer Science and Network Security* 7(9), 116–122 (2007)
- [10] Taqqu, M., Teverovsky, V., Willinger, W.: Estimators for Long-Range Dependence: An Empirical Study. *Fractals* 3(4), 785–798 (1995)

- [11] Taqqu, M.S., Teverovsky, V.: Robustness of Whittle-type Estimates for Time Series with Long-Range Dependence. *Stochastic Models* 13, 723–757 (1997)
- [12] Patrice, A., Darryl, V.: Wavelet Analysis of Long-Range-Dependence Traffic. *IEEE Transactions on Information Theory* 44(1), 2–15 (1998)
- [13] Stoev, S., Taqqu, M., Park, C., et al.: On the Wavelet Spectrum Diagnostic for Hurst Parameter Estimation in the Analysis of Internet Traffic. *Computer Networks* 48(3), 423–445 (2005)
- [14] David, R., Flaminio, M., Sebastia, S.: Segmenting LRD Traffic with Wavelets and the Schwarz Information Criterion. In: 2006 Passive and Active Measurements Conference, pp. 121–130. IEEE Press, Adelaide (2006)
- [15] Li, M.: Error Order of Magnitude for Modeling Autocorrelation Function of Interarrival Times of Network Traffic Using Fractional Gaussian Noise. In: 7th WSEAS International Conference on Applied Computer & Applied Computational Science, pp. 167–172. IEEE Press, Venice (2008)

User-Assisted Host-Based Detection of Outbound Malware Traffic*

Huijun Xiong¹, Prateek Malhotra¹, Deian Stefan², Chehai Wu³, and Danfeng Yao¹

¹ Department of Computer Science, Rutgers University Piscataway, NJ 08854, USA
huijun@cs.rutgers.edu, someone1@eden.rutgers.edu,
danfeng@cs.rutgers.edu

² Department of Electrical Engineering, The Cooper Union, New York, NY 10003, USA
stefan@cooper.edu

³ AppFolio, Inc. 55 Castilian Dr. Goleta, CA 93117, USA
wuchehai@gmail.com

Abstract. Conventional network security solutions are performed on network-layer packets using statistical measures. These types of traffic analysis may not catch stealthy attacks carried out by today's malware. We aim to develop a host-based security tool that identifies suspicious outbound network connections through analyzing the user's surfing activities. Specifically, our solution for Web applications predicts user's network connections by analyzing Web content; unpredicted traffic is further investigated with the user's help. We describe our method and implementation as well as the experimental results in evaluating its efficiency and effectiveness. We describe how our studies can be applied to detecting bot infection. In order to assess the workload of our host-based traffic-analysis tool, we also perform a large-scale characterization study on 500 university-users' wireless network traces for 4-month period. We study both the statistical and temporal patterns of individuals' web usage behaviors from collected wireless network traces. Users are classified into different profiles based on their web usage patterns. Our results show that users have regularities in their Web activities and the expected workload of our traffic-analysis solution is low.

1 Introduction

Several studies estimate that millions of computers worldwide are infected by malware and have become bots that are controlled by cyber criminals [9,10,14]. The infected computers are coordinated and used by the attackers to launch diverse malicious and illegal network activities, including perpetrating identity theft, sending spam (estimated 100 billion spam messages every day [25]), launching denial of service (DoS) attacks, and committing click fraud. Malicious bots are stealthy and difficult to detect using conventional anti-virus software. Botnet communications including command and control (C & C) and attacks disturb the usual and routine traffic patterns of a user. For a good description of the botnet structures, we refer readers to the paper by Dagon, Gu, Lee,

* This work has been supported in part by NSF grant CCF-0728937, CNS-0831186, and the Rutgers University Computing Coordination Council Pervasive Computing Initiative Grant.

and Lee [6]. Malicious bot is a special type of malware, which is an umbrella term for all malicious software such as virus, worm, rootkit, and spyware.

Many network-wide intrusion detection and protection systems, both commercial products or research prototypes, have been developed for monitoring network traffics and report alerts if observing known suspicious attack patterns [24,26]. Similarly, anomaly detection systems aim to detect deviations or abnormal events from historic usage patterns [22]. However, the existing network analysis and security tools are inadequate in two main aspects: *individualized analysis* and *personalized security*. It is reported that on average 3-5 percent of organizational assets are compromised by bots and malware – even when the best and most up-to-date security software is applied [7]. Most current network trace analysis focuses on the aggregated traffic flow of the entire network, e.g., network-side traffic volume, busiest hosts on the network, and bursty periods of the organization. These types of network-wide traffic analysis do not give insights to the usage patterns of individuals on the network.

In this paper, we describe a novel host-based anomaly detection approach based on both traffic prediction and user participation. We call it a *personalized security* approach. Specifically, we implement a traffic monitoring framework that is capable of *predicting* legitimate outbound network connections. Our framework intercepts network traffic from the host. The connections that are observed but *not* predicted by the framework may be due to malware activities on the host.

In order to further classify the unpredicted outbound connections, our approach is to leverage the user's personal knowledge about his or her Web activities, for example, by prompting a window asking the user whether she initiated a connection to a Web address. Studies found that users demonstrated regularities in their surfing patterns [12]. Our characterization results presented in this paper also indicate that users have highly repetitive network-connection patterns. Many botnets successfully evade the IRC (Internet Relay Chat protocol)-based detection by switching to HTTP-based command and control [13,30], as HTTP traffic is usually allowed through firewalls and not blocked.

Therefore, our study focuses on identifying HTTP traffic of malware. Our approach can be generalized to other application protocols. With a personalized security approach, we monitor and examine host-based traffic patterns to detect abnormal network requests caused by malware. This type of investigations represents a personalized analytical approach that can also be applied to managing the security of large organizations. We implement our traffic-monitoring framework in Python and evaluate its efficiency and prediction effectiveness. The technical challenges involved in predicting Web-related traffic are the diversity and flexibility of hypertexts including scripts. We focus on parsing and analyzing static Web pages, embedded iframes and cascaded style sheets, as well as redirected pages. *Our results indicate that most traffic can be effectively predicted using our code for static Web content.* Legitimate connections that our prediction misses are mainly due to JavaScript code. In an effort to reduce the number of questions asking to the user, we also utilize a whitelisting approach.

Another contribution of this paper is a large-scale characterization study on 500-users' wireless network traces for four-month period. We collected wireless network traces from a university. Our characterization work aims towards discovering the patterns and properties of individuals' network behaviors, in particular, we study both the statistical and temporal patterns of individual host's Web activities. (A host is uniquely identified by its MAC address which remains consistent throughout the dataset.) Our investigation is different from the conventional network-wide aggregated traffic analysis, as we focus on the micro-scale pattern of an individual user. Our characterization results suggest that users have low diversity in terms of daily Web sites visited – people tend to visit a small number of Web sites regularly. This repetitiveness can be leveraged to construct effective host-based malware detection solutions because malware-caused deviations from the regular patterns may be identified. It also indicates that the expected workload of our traffic-monitoring tool is low.

The rest of the paper is organized as follows. Our host-based traffic-monitoring framework is described in Section 2. Our wireless network trace analysis is given in Section 3. The related work is described in Section 4. Conclusions and future work are in Section 5.

2 Outbound Malware-Traffic Detection with User Participation

In this section, we describe a traffic monitoring framework that aims to identify malware traffic by carefully analyzing user's Web requests and content as well as involving the user in the process of classifying traffic. Although detecting anomaly traffic with user's help may appear to be straightforward, the challenge here is how to encourage user's participation and avoid intrusiveness to user. Thus we need a precise and efficient prediction mechanism. Our solution requires the minimal participation from the user and causes no undesirable delays to the user's surfing experience. Our study is focused on Web traffic because HTTP based malware activities such as Spyware or botnet command and control are notorious hard to detect – most firewalls allow HTTP traffic on port 80. Our implementation is realized in Python in Linux, but the architecture can be realized in other programming languages and platforms as well.

Our malware attack model and security assumptions are as follows. We consider stealthy malware that is secretly sending outbound HTTP traffic. The malware may corrupt the browser, e.g., through malicious extensions [16]. Thus, the browser is *not* assumed to be trusted. The malware may be at the application-level or kernel-level such as rootkits which actively hide their presence from the host's operating system. However, for kernel-level malware, we assume that components in our detection framework along with its files are not corrupted by the malware. This last assumption is reasonable, as the integrity of our framework can be ensured using trusted computing infrastructure such as Trusted Platform Module (TPM) [28,29] that are available on most commodity PCs through a standard attestation procedure [20]. The integration of TPM into our framework is not described in this paper. Our study addresses client-side security, and complements any server-side security solutions.

2.1 System Architecture and Algorithm for Traffic Monitoring

In order to identify unauthorized HTTP connections possibly due to malware, our approach is to monitor and analyze outbound network requests. Blocking outbound malware packets can effectively render malware useless. Thus, we do not need to examine all incoming traffic, which makes our solution all the more efficient. Our solution can effectively eliminate a wide spectrum of harmful malware activities, e.g., identity theft, spam, DDoS attacks, click fraud, or botnet command & control messages. Malware is unable to deliver stolen personal data to the outside. Our framework has the following three main components:

- *Sniffer*: intercepting and filtering all outbound HTTP requests. Pending outbound HTTP requests are put on a list waiting for approval to execute. Sniffing outbound HTTP requests can be realized using existing network libraries such as `libpcap` library in Python.
- *Predictor*: the prediction of legitimate outbound HTTP requests based on the user's activities and parsing of Web content retrieved *out of band* (Section 2.2). Observed-yet-unpredicted connections will be prompted to the user for further classification. An important requirement to the predictor is to reduce the number of questions for the user while maintaining the prediction accuracy.
- *User interface*: pop-up windows where a user can indicate whether or not observed network attempts are initialized by her (Section 2.3). The interface needs to be easy to use by nontechnical-savvy users. A screenshot of our user interface is shown in Figure 3.

In the next few sections, we will describe the technical details involved in realizing our predictor as well as our experimental evaluation on the framework.

2.2 Analysis on Web Contents

In HTTP protocols, each object is retrieved in a separate HTTP request. For example, if a Web page has 10 images, then the browser issues 11 separate HTTP requests sequentially to the Web server. For persistent HTTP connections, all 11 HTTP requests may be sent in one TCP connection between the server and the client, whereas for nonpersistent HTTP connection, each HTTP request requires a separate TCP connection. Being persistent or not does not affect the deployment of our solution.

To predict legitimate Web traffic, a straightforward solution is that each time an outbound HTTP request is observed, we ask the user whether she is responsible for that connection. However, this simple approach may create many questions and be quite intrusive to users due to the pervasive third-party content on the Web – advertisements or (multimedia) content hosted by content delivery providers instead of the main web server. Third-party content (e.g., from amakai.com or ying.com) is retrieved from URLs that may seem arbitrary to the user, i.e., bearing no similarity to the main website URL (e.g., yahoo.com), impacting user's classification decisions. This problem is solved by us with out-of-band retrieval and analysis of Web content (explained below). The workflow for identifying suspicious outbound traffic in our solution is as follows.

1. The predictor fetches the requested Web page independent of the browser (e.g., using `wget`), which we call *out-of-band retrieval*. It parses the retrieved content to whitelist the outbound HTTP requests for fetching referenced objects (e.g., images). The whitelist is stored in memory, and is domain-based to improve our prediction efficiency.
2. The sniffer intercepts all attempted outbound HTTP connections from the host (including those from applications other than the browser), which are put into a waiting list. The HTTP requests that appear on the predictor's whitelist are permitted. For example, a HTTP GET request to fetch object `ying.com/images/tree.jpg` is allowed if `ying.com` is on the whitelist.
3. For the pending connections that cannot be predicted, we prompt a small window to the user asking whether she has initialized that request. The connection is allowed if the user enters *Yes*, and denied otherwise.

A schematic drawing of the detailed workflow regarding our traffic prediction is shown in Figure 1 and explained as follows. Figure 2 gives an example of the objects/connections predicted as a result of a user visiting `www.cs.rutgers.edu`.

1. The user visits a target website W in Step 1 and 2. This initial request to URL W is intercepted by `libpcap` library in Step 3.
2. In Step 4, our predictor checks to see if the domain of W is whitelisted or not. If the domain is blacklisted, then the user is given a warning. If it is whitelisted, the request is allowed. Otherwise, we prompt the user to confirm URL W as shown in Figure 3 in Step 5. User-permitted domains are put onto the whitelist for future reference.
3. In Step 6, the object `HTTPRedirectHandler` is for keeping track of redirected requests by putting a listener to each executed outbound HTTP request. Therefore, our predictor is capable of tracking redirections of any arbitrary depth. We note that our analysis including the sending and processing of HTTP requests is outside and independent of the browser, which we call *out-of-band analysis*.

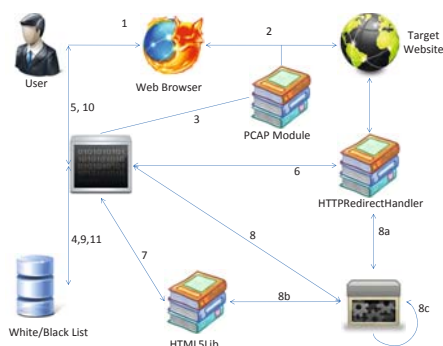


Fig. 1. Workflow in our traffic-monitoring framework

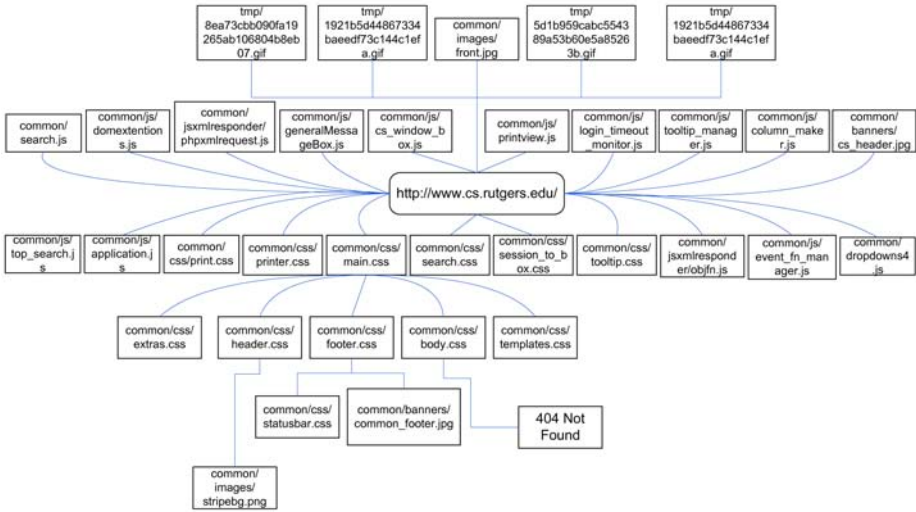


Fig. 2. The illustration of a tree capturing the hierarchical invoking sequences among (automatic) outbound HTTP requests as a result of visiting www.cs.rutgers.edu

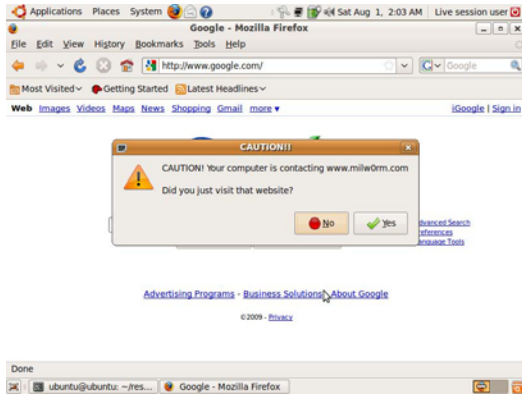


Fig. 3. A screenshot of our user interface

4. The core parsing and prediction steps are Step 7 and 8 (8a, 8b, and 8c). Content retrieved by the out-of-band request is parsed by the HTML5Lib Python library to predict additional HTTP requests for objects therein. For example, in Figure 2, .CSS file may contain additional image objects that need to be requested. This process loops (indicated by Step 8c) until there is no more object to retrieve.
5. Unique domain names of predicted connections are put on the whitelist in Step 9. Observed actual connections that do not appear on the whitelist are prompted to the user for classification in Step 10, the results of which are used to update the white/black lists in Step 11.

In our current implementation, our analysis is in parallel with the browser and takes place in a *postmortem* fashion where we aim to identify suspicious URL connections and alert to the user. Therefore, it causes no delay in the user's actual surfing experience. An alternative solution is to suspend all outbound HTTP requests until they are either predicted by our tool or explicitly approved by the user. This approach may cause delays in users' Web usage and is not adopted by us.

Because our analysis is completely independent of the browser, it is robust against corrupted browser. For example, a browser with malicious extensions on the client may secretly exporting users' personal data to the attacker (e.g., spyware); our solution can detect the stealthy traffic. Our parser utilizes the `HTML5LIB` Python module. This module provides a robust ability to parse through HTML/XML and XHTML code including those with malformed markup code. The module can automatically fix bad markup and return the parsed data in several formats including a tree format. After we download and parse through an HTML file, our predictor goes through every node in the tree using a built-in tree-traversal mechanism from the `HTML5lib` module to identify tags with the attribute `SRC`. HTML tag with the attribute `SRC` initiates a network request to fetch the contents of the source destination. The tags include `IMG`, `SCRIPT`, `IFRAME`, `FRAME`, and `INPUT`, as well as tags `AUDIO`, `EMBED`, `VIDEO` in HTML5. The content found within style tags are parsed through for any URL `includes`. This procedure predicts `@import` requests and image `includes` for backgrounds or behavior scripts. Last but not the least, attribute `HREF` from link tags is parsed, as it usually contains the `include` for CSS files. In essence, we recursively identify objects referred in retrieved WebPages to estimate the (separate) HTTP connections required to fetch all of them. We note that our solution does not require crawling hyperlinks and thus is quite efficient. Advanced Web contents such as Applets or AJAX (asynchronous JavaScript and XML) requests through `XMLHttpRequest` typically concern objects residing on the same domain as their parent page, and thus are safely disregarded by us.

2.3 Experimental Evaluation

In this section, we describe experimental evaluation on our solution. All the experiments were executed on a HP Pavilion dv9500t laptop computer that has 4GB memory, an Intel Core 2 Duo 2.2GHz CPU with ArchLinux X86_64.

In Table 1, we evaluate our program on several websites to assess its ability to predict legitimate outbound HTTP connections. For most websites studied, our program is able to predict most of the actual requests. For websites with the heavy use of JavaScript code such as digg.com, the prediction percentage is relatively low. Improving our prediction on JavaScript-generated requests requires interpreting JavaScript code along with the DOM object *out of the browser*. This task is subject to our future study.

We perform extensive experiments to evaluate the efficiency of the *predictor* mechanism in our implementation. The prediction is performed in parallel with the actual Web requests by the browser, and thus its execution has little impact on the browser's responsiveness to the user. Nevertheless, fast prediction is desirable because of the early detection of suspicious outbound HTTP requests. We evaluate four websites with distinct characteristics and our results shown in Table 2. blogs.zdnet.com is a very dynamic and rich website, which usually has new connections on every

Table 1. Evaluation on the prediction ability of outbound HTTP requests. Req. stands for requests. Dom. stands for domains.

URL	Actual Req.	Predicted Req.	% Predicted	Actual Dom.	Predicted Dom.
yahoo.com	37	67	92%	7	4
eset.com	51	67	94%	5	2
google.com	7	3	43%	3	2
cs.rutgers.edu	34	39	100%	1	1
digg.com	111	47	42%	21	7
codeigniter.com	29	177	100%	1	6

Table 2. Evaluation of prediction efficiency on four websites with full (F.) or lite (L.) predictors. Results are averaged from three runs. The time is shown in seconds.

	blogs.zdnet.com	www.cs.rutgers.edu	yahoo.com	google.com
Actual Req.	228	67	40	3
Network time (F.)	23.50	3.54	1.86	0.22
Parsing time (F.)	2.19	0.28	0.09	0.03
Total (F.)	30.99	4.035	2.87	0.28
Network time (L.)	3.75	0.22	0.56	0.19
Parsing time (L.)	0.83	0.27	0.09	0.03
Total (L.)	12.55	0.68	1.39	0.24

page load. yahoo.com provides less dynamic content than blogs.zdnet.com – this Web page stays consistent over a short period of time (e.g., a couple of days). www.cs.rutgers.edu is a static and medium-sized website. google.com is a very light and mostly static website. It only uses JavaScript code for the pull-down menu at the top.

We test two versions of our predictor implementation: a *full* predictor and a *lite* predictor. The full predictor is as described in Section 2.2, where each requested object is analyzed in the same fashion. In the lite predictor, images and JavaScript objects are not requested and processed, i.e., Steps 6 through 8c are skipped if the request is to fetch an image or JavaScript object. The lite version is effective for most websites and may significantly improve our prediction efficiency. For yahoo.com and www.cs.rutgers.edu, the lite predictor is faster than the full predictor, in particular for the more dynamic pages. For simple static page like google.com, the two versions do not differ much as expected. For blogs.zdnet.com, prediction time goes down with the lite predictor. However, some unique domains are not discovered in the process: certain objects are not found (404) or being redirected (300); these cases are not pursued by the lite predictor. Our experimental results indicate that both the full and lite versions of predictors perform reasonably well for typical websites on a personal computer.

3 Analysis on University Wireless Network Traces

Our solution provides a real-time suspicious out-bound traffic discovery mechanism. In detection phrase, we need user's participation to classify suspicious traffic into malicious or good traffic. In order to assess user's workload of our host-based traffic-analysis tool, we carry out a characterization study on 500 university-users' wireless network traces for 4-month period. We study both statistical and temporal patterns of *individuals'* Web usage behaviors from collected wireless network traces. Unlike pattern recognition based detection mechanism, we use these patterns to gain an insight in user's network activity, which implies the user's workload in using our system.

We run several filters on the original data. First, we remove the users whose total traffic volume is lower than 1 MB, which effectively remove the users with failed login and temporary users. Second, we only keep the outgoing and incoming TCP traffic with destination or source port 80, in order to filter out non-HTTP connections and exclude data from peer-to-peer software. Many HTTP-based P2P applications run on high port numbers. We find that there are many invalid MAC addresses in our data. We wrote a program to automatically verify the validity of a MAC address by comparing it with the published prefixes of authorized network interface card manufactures. Unmatched MAC addresses are notified and their corresponding traffic is removed.

In what follows, we use the words *host* and *user* interchangeably, as the hosts that connect to the wireless network are virtually all personal laptops.

Volume of Distinct IP Addresses. Our overall analysis methodology is to explore and characterize network activities belonging to individual hosts. We compute and categorize each host's daily web traffic volume. We choose the top 500 users represented by distinct local MAC addresses that have the highest number active days. In an inactive day, the host has zero HTTP traffic with port 80. In Figure 4 (left), hosts are categorized by their daily numbers of IP addresses visited. Y-axis denotes the number of users who are in a category represented by the values in the square bracket. The majorities of users out of the 500 studied only visited a small number of servers and have low diversity in their daily Web traffic. On the other hand, a few users are extremely active and visit a large number of distinct IP addresses every day. On average, 278 hosts contacted less than 50 distinct IP addresses daily. Note that duplicate HTTP connections are counted only once, thus automatic refresh or reload operations by Web servers do not artificially increase the count of IP addresses.

Temporal Analysis Of Individual Web Usage. We analyze how many *new* IP addresses and *old* IP addresses that a host visits each day. If an IP address visited by a host exists in the previous surfing history, then it is labeled as an *old IP*, otherwise, a *new IP* of that day. The surfing history of a user is initialized to be empty on day one, i.e., $H_1 = \emptyset$. Thus, all traffic on that day is new. At the beginning of an active day i , the surfing history is concatenated with day $i - 1$'s new IP set new_{i-1} , i.e., $H_i = H_{i-1} \cup new_{i-1}$. Thus, an IP address at day i is new only with respect to a user's surfing history up to day i . We observe that most hosts visit fewer numbers of new IP addresses than old IP addresses each day, indicating that most nodes are consistent with their surfing history. In Figure 4 (right), X-axis is the index of each user, Y-axis is daily number of IP addresses that visited by a user, the solid line is daily number of old IP addresses, and the dotted one

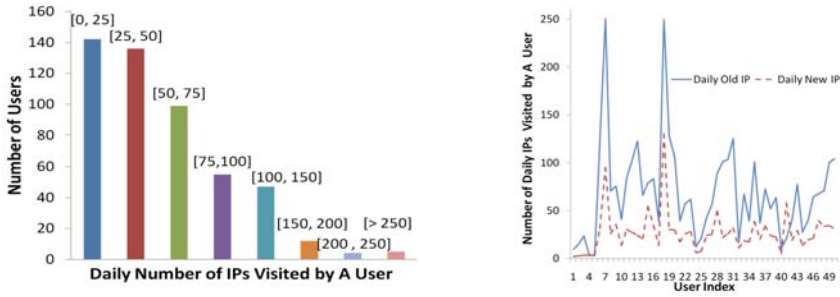


Fig. 4. Left: The number of hosts (in Y-axis) based on their volume of daily distinct IP addresses visited. Values in square brackets are the ranges of daily number of IP addresses visited. Right: The numbers of daily old and new IP addresses visited by 50 hosts, respectively.

is the daily number of new IP addresses. We select the most active 50 hosts to show in this analysis. For most users in Figure 4, the number of new IP addresses grows with the number of old IP addresses. An overwhelming majority of hosts visited many more old IP addresses than new IP addresses on active days.

We show the analysis results on two specific hosts in Figure 5, where Y-axis denotes the percentage of daily old IP addresses visited by a host and X-axis denotes the index of active days. For each day, the percentage is computed as the number of old IP addresses divided by the number of all IP addresses visited by the host. Inactive days are not included in the analysis. Both hosts visit significantly more old IP addresses than new ones each day. Both users demonstrate repetitive patterns in their surfing history. As the volume of new IP addresses is significantly lower than that of old IP addresses, the workload for analysis and monitoring would be low. For our host-based bot detection approach described in Section 2, we aim to focus on analyzing new IP addresses or URLs visited by a host.

We also identify the two active hosts in our dataset and find that both hosts have a high degree of repetitiveness in the IP addresses visited. In Table 3, we count the

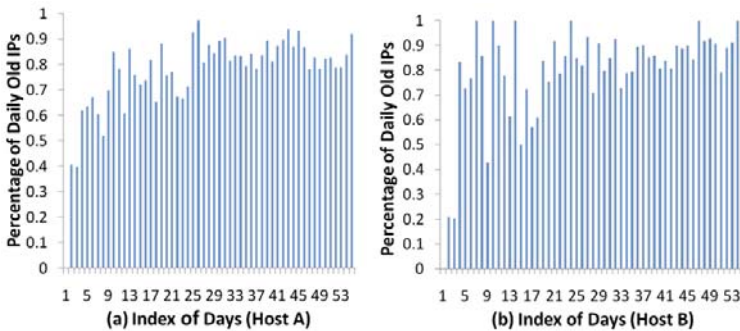


Fig. 5. Percentage of daily old IP addresses visited by two hosts, respectively. Both hosts have 55 active days.

Table 3. Numbers of days during which two hosts visit a certain percentage of old IP addresses, respectively. Total numbers of active days are 122 and 118, respectively.

Percentage of Old IP addresses Visited	Host 1	Host 2
100%	59 days	57 days
$\geq 90\%$	80 days	79 days
$\geq 80\%$	100 days	95 days

numbers of days during which two hosts visit a certain percentage of old IP addresses, respectively. For each active day of a host, the percentage is computed as the number of old IP addresses visited divided by the total IP addresses visited during that day.

Profiling Visit Patterns Of Individual Hosts. The degree of activity of a host can be represented by the number of total and new IP addresses visited daily or the number of active days during the 4-month long period. Intuitively, a user who surfs on Internet regularly tends to be more proficient in using the web and visit a diverse and large number of IP addresses. In an effort to investigating the correlation between the two metrics, we plot the number of daily visited IP addresses against the number of active days of a user in Figure 6. We study the 500 most active hosts whose numbers of active days range from 10 to 130 days. In Figure 6, X-axis in each graph is the number of active days; Y-axis is the daily number of the total IP addresses in (a) and new IP addresses in (b), respectively. We find that for some hosts the number of IP addresses visited grows with the number of active days, which is consistent with our intuition. The area close to the origin is dense with points in Figure 6 indicating that the majority of users studied have limited HTTP-based network activities both in terms of active days and the volume of distinct remote IP addresses visited. We further draw a 3-D plot with 500 hosts according to their (1) number of active days (2) daily IP addresses visited and (3) daily new IP addresses visited. The details can be found in technical report [32].

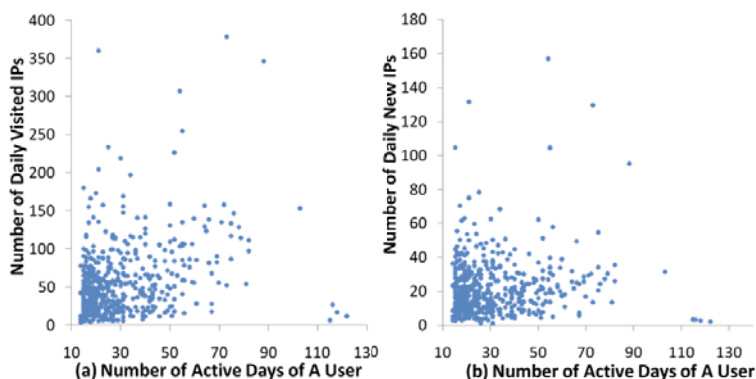


Fig. 6. X-axis is the number of active days of a host. Y-axis is the average number of the total IP addresses in (a) and new IP addresses in (b), respectively. 500 hosts are plotted. Each dot represents a user.

Summary on wireless network traces. The number of distinct IP addresses visited by a host is most likely to be higher than the actual number of websites visited. A website refers to the top-level domain name, e.g., `google.com`, `yahoo.com`, `amazon.com`. There are several reasons. (1) Many popular servers have multiple IP addresses for load-balancing and fault-tolerance purposes. For example, `066.102.001.166` and `074.125.067.118` are among the several `google.com` servers. They are counted as two different IP addresses in our analysis. (2) Many websites heavily use third-party content providers for multimedia contents or advertisements. For example, when `www.cnn.com` is loaded, several content providers are contacted. (3) Dynamic IP addresses due to DHCP cause a single (remote) host to have distinct IP addresses at different time. More fine-grained data collection and analysis on the URLs and websites visited by the users are subject to our future study. We note that our analysis is based on distinct IP addresses visited as opposed to URLs, due to the limitation of our dataset. As a Web server typically hosts many Web pages, the actual number of URLs visited by the user may differ from the number of IP addresses.

4 Related Work

Our proposed personalized security approach is different from existing anomaly detection techniques [3][22]. Personalized security aims at exploring individuals' and personal usage patterns for the detection purpose, whereas conventional anomaly detection methods construct generic solutions for users. In addition, we focus on detecting *internal* malware abuse and threats, as opposed to preventing break-ins coming from outside in the conventional settings. Security-oriented traffic analysis has caught much attention from both the network and security communities, including malware or botnet characterization [8][21][23][31] and privacy-preserving routing and packet trace anonymization [17][18][19]. Our work differs from them in that i) we analyze the statistical and temporal patterns of *individuals'* application-layer Web usage (as opposed to low-level network packets); and ii) our analysis is user-centric by leveraging user's personal knowledge about her own surfing activities. To that end, our solution aims to address the usability, in particular nonintrusiveness, of the host-based malware detection solutions. Our tool is complementary to the existing network-level or program-level malware identification solutions.

Analyzing and characterizing organizational wireless network traces have traditionally been studied for maintaining the stability and availability of network resources [4]. Several studies have been performed on university campus wireless network traces [1][11][15][27]. Researchers in Stanford University [27] studied a 12-week trace of their local-area wireless network in Computer Science Department building with attempt to find out the peak throughput rates and the cause of the peaks. Kotz and Essien [15] carried out a similar study with a significantly larger and broader population. They found out that network backup and file-sharing traffic contributed an unexpectedly large amount to the overall traffic, which was also found in [11].

Authors in the paper [12] paid more attention on user behavior in the wireless network, but these behavior studies served as parameters for network performance optimization. For example, authors in [1] pointed out the load of each access point was

determined mostly by individual user behaviors. In [2], it was found that the data-transfer rates of users follow a power law distribution. Power-law distributions were also found in certain characteristics of WWW, such as the distributions of document sizes and user requests for documents [5]. In comparison, our study on user behaviors in local wireless network differs from aforementioned studies. As opposed to optimizing network or server performance, we focus on analyzing users' individual and temporal behavior patterns in their wireless Web traffic, which is motivated by the need for personalized security. Policies in firewalls are typically based on port numbers and IP addresses. In contrast, we provide much more fine-grained inspection as we examine each HTTP connection intercepted on the network interface.

5 Conclusions and Future Work

In this paper, we proposed a novel host-based security tool that identifies suspicious outbound network requests with user's participation. Specifically, we described a personalized security approach and a simple-yet-effective host-based network security solution that identifies abnormal outbound HTTP requests based on *out-of-band* (i.e., browser-independent) prediction and user-assisted classification. We also described our results on analyzing a large-scale wireless network dataset that involves more than 500 users over 4-month period. We analyzed the *individual* usage patterns of users in an organization in order to assess the workload of our host-based malware detection solution. Our characterization analysis on individuals' surfing patterns is useful beyond the specific malware-detection problem studied, as it provides insights to how individual surfing-behavior patterns may be leveraged for improved web services.

For future work, we plan to carry out user studies to evaluate humans' traffic recognition abilities. The hypothesis that we aim to evaluate in the user study is that a user knows the websites she is currently visiting and thus can recognize malware-related traffic to unfamiliar URLs. In a user study, each participant will be asked to freely surf online for 10 to 20 minutes, during which we will randomly access a list of arbitrary (bot) servers, i.e., inject malware traffic to test whether a user can recognize it. For unpredicted outbound HTTP requests including the injected ones, we will prompt a window (as in Figure 3) asking whether or not the user just visited the URL. From users' responses, we will compute false positive and false negative rates of their performance. Here, a false positive result will indicate that the user misclassified legitimate user-initiated traffic as malware HTTP requests. A false negative result, conversely, will indicate that the participant has misclassified bot URLs for their own traffic. With our comprehensive traffic prediction mechanism described in this paper, we expect this security tool to be nonintrusive to users.

In addition, we will investigate techniques to include personalized semantic analysis of surfing records and novel clustering methods for identifying outliers and suspicious traffic. This study will first extract *surfing tastes* of individuals and then detect suspicious Web requests whose content is *inconsistent* with the user's previous surfing history. We also plan to construct more advanced pattern-recognition techniques on individuals' usages.

References

1. Balachandran, A., Voelker, G.M., Bahl, P., Rangan, P.V.: Characterizing User Behavior and Network Performance in A Public Wireless LAN. In: SIGMETRICS 2002: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 195–205. ACM, New York (2002)
2. Balazinska, M., Castro, P.: Characterizing Mobility and Network Usage in A Corporate Wireless Local-Area Network. In: MobiSys 2003: Proceedings of the 1st international conference on Mobile systems, applications and services, pp. 303–316. ACM, New York (2003)
3. Borders, K., Prakash, A.: Web Tap: Detecting Covert Web Traffic. In: Atluri, V., Pfizmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 110–120. ACM, New York (2004)
4. A Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD), <http://crawdad.cs.dartmouth.edu/>
5. Cunha, C., Bestavros, A., Crovella, M.: Characteristics of WWW Client-based Traces. Technical report, Boston, MA, USA (1995)
6. Dagon, D., Gu, G., Lee, C.P., Lee, W.: A Taxonomy of Botnet Structures. In: ACSAC, pp. 325–339 (2007)
7. Emerging Cyber Threats Report for 2009, Georgia Tech Information Security Center (October 2008)
8. Gianvecchio, S., Xie, M., Wu, Z., Wang, H.: Measurement and Classification of Humans and Bots in Internet Chat. In: Proceedings of USENIX Security Symposium (2008)
9. Gu, G., Perdisci, R., Zhang, J., Lee, W.: Botminer: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In: Proceedings of the 17th USENIX Security Symposium (2008)
10. Gu, G., Zhang, J., Lee, W.: Botsniffer: Detecting Botnet Command and Control Channels in Network Traffic. In: Proceedings of the 15th Annual Network and Distributed System Security Symposium, NDSS (2008)
11. Henderson, T., Kotz, D., Abyzov, I.: The Changing Usage of A Mature Campus-wide Wireless Network. In: MobiCom 2004: Proceedings of the 10th annual international conference on Mobile computing and networking, pp. 187–201. ACM, New York (2004)
12. Huberman, B.A., Pirolli, P.L., Pitkow, J.E., Lukose, R.M.: Strong Regularities in World Wide Web Surfing. *Science* 280(95) (1998)
13. Ianneli, N., Hackworth, A.: Botnets as A Vehicle for Online Crime (2005), <http://www.cert.org/archive/pdf/Botnets.pdf>
14. Karasaridis, A., Rexroad, B., Hoeflin, D.: Wide-Scale Botnet Detection and Characterization. In: HotBots 2007: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, Berkeley, CA, USA, p. 7. USENIX Association (2007)
15. Kotz, D., Essien, K.: Analysis of A Campus-wide Wireless Network. In: Proceedings of ACM Mobicom, pp. 107–118. ACM Press, New York (2002)
16. Louw, M.T., Lim, J.S., Venkatakrisnan, V.N.: Enhancing Web Browser Security Against Malware Extensions. *Journal in Computer Virology* 4(3), 179–195 (2008)
17. Mogul, J.C., Arlitt, M.: SC2D: an Alternative to Trace Anonymization. In: MineNet 2006: Proceedings of the 2006 SIGCOMM workshop on Mining network data, pp. 323–328. ACM, New York (2006)
18. Pang, R., Allman, M., Paxson, V., Lee, J.: The Devil and Packet Trace Anonymization. *SIGCOMM Comput. Commun. Rev.* 36(1), 29–38 (2006)
19. Pang, R., Paxson, V.: A High-level Programming Environment for Packet Trace Anonymization and Transformation. In: SIGCOMM 2003: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 339–351. ACM, New York (2003)

20. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and Implementation of a TCG-based Integrity Measurement Architecture. In: USENIX Security Symposium, pp. 223–238. USENIX (2004)
21. Saroiu, S., Gribble, S.D., Levy, H.M.: Measurement and Analysis of Spyware in a University Environment, pp. 141–153
22. Shieh, S.-P., Gligor, V.D.: On a Pattern-Oriented Model for Intrusion Detection. *IEEE Transactions on Knowledge and Data Engineering* 9(4) (July/August 1997)
23. Singh, S., Estan, C., Varghese, G., Savage, S.: Automated Worm Fingerprinting. In: OSDI 2004: Proceedings of the 6th conference on Symposium on Operating Systems Design Implementation, Berkeley, CA, USA. USENIX Association (2004)
24. SNORT, an open source network intrusion prevention and detection system, <http://www.snort.org/>
25. Stewart, J.: Top Spam Botnets Exposed (April 2008), <http://www.secureworks.com/research/threats/topbotnets>
26. Symantec, <http://www.symantec.com/index.jsp>
27. Tang, D., Baker, M.: Analysis of A Local-Area Wireless Network. In: *MobiCom 2000: Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 1–10. ACM, New York (2000)
28. Trusted Computing Group. Trusted platform module main specification, Part 1: Design principles, Part 2: TPM structures, Part 3: Commands. Version 1.2, Revision 62 (October 2003)
29. TCG PC Client Specific TPM Interface Specification (TIS), Version 1.2. Trusted Computing Group, http://www.trustedcomputinggroup.org/groups/pc_client/
30. Webb, S., Caverlee, J., Pu, C.: Predicting Web Spam with HTTP Session Information. In: *Proceedings of the Seventeenth Conference on Information and Knowledge Management (CIKM 2008)* (October 2008)
31. Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulthen, G., Osipkov, I.: Spamming Botnets: Signatures and Characteristics. In: *SIGCOMM 2008: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pp. 171–182. ACM, New York (2008)
32. Xiong, H., Malhotra, P., Stefan, D., Wu, C., Yao, D.: User-Assisted Host-Based Detection of Outbound Malware Traffic. Technical report, Rutgers University (October 2009)

Assessing Security Risk to a Network Using a Statistical Model of Attacker Community Competence

Tomas Olsson

Swedish Institute of Computer Science
P.O. Box 1263, SE-164 29 Kista, Sweden
tol@sics.se

Abstract. We propose a novel approach for statistical risk modeling of network attacks that lets an operator perform risk analysis using a data model and an impact model on top of an attack graph in combination with a statistical model of the attacker community exploitation skill. The data model describes how data flows between nodes in the network – how it is copied and processed by softwares and hosts – while the impact model models how exploitation of vulnerabilities affects the data flows with respect to the confidentiality, integrity and availability of the data. In addition, by assigning a loss value to a compromised data set, we can estimate the cost of a successful attack. The statistical model lets us incorporate real-time monitor data from a honeypot in the risk calculation. The exploitation skill distribution is inferred by first classifying each vulnerability into a required exploitation skill-level category, then mapping each skill-level into a distribution over the required exploitation skill, and last applying Bayesian inference over the attack data. The final security risk is thereafter computed by marginalizing over the exploitation skill.

1 Introduction

In order to manage the dynamic nature of the security of a network, where new nodes are added and new softwares are installed, operators regularly run scanning tools such as Nessus to discover the network topology and existing vulnerabilities [1]. However these tools do not put vulnerabilities into the context of how these vulnerabilities can be exploited to obtain illegal access to resources in the network. Likewise, they do not automatically determine the impact to the system or the potential risk.

As a consequence, a great number of automated approaches to security analysis have been proposed during the last decade [2]. Some of these automated approaches define security metrics over the paths of an attack graph [3,4,5,6,7], while others define a security metric in terms of security risk [8,9,10,11].

In this paper, we follow the second path by proposing a novel approach to compute the security risk. Risk is usually defined as the expected loss or as defined in [12]:

Risk is a function of the likelihood of a given threat-source's exercising a particular potential vulnerability, and the resulting impact of that adverse event on the organization.

Accordingly, a risk computation consists of two parts: a probability estimation of a successful attack and an impact estimation.

Most of the papers referred to above, use a simple model for estimating the impact to the network. Typically, they either do not use an explicit model, e.g. only probabilities, or sum the assigned weights of compromised nodes. As an example, in [13], the security of two networks, one patched and one unpatched, was compared by computing the number of hosts removed from the attack graph or by assigning a weight of importance to each host as well. This would be equivalent to estimating the impact by counting the number of compromised hosts in the attack graph or by computing the sum of their weights. In addition, many of the papers that use probabilistic models do not describe of how to estimate these probabilities. Therefore, in this work, we extend existing work with a more advanced network impact model and with a novel approach for inferring probabilities over attack graphs using a Bayesian approach assuming that a record of historical attack data from a honeypot is given [14].

The network impact model consists of a data model in combination with an impact model over an attack graph. The data model models how data is copied between softwares in a network, thereby taking into account the dependencies between different hosts and the dependencies between different data flows. The impact model describes what impact individual vulnerabilities have on the data sets with respect to confidentiality, integrity and availability as well as the impact propagation from compromised data sets. By assigning loss values to data sets, we can estimate the cost of a successful attack. Thus, in our work, an operator does not need to understand the importance of each node in a network as is the case in the papers referred to above.

The probability estimation of a successful attack is computed by first classifying each vulnerability into a required exploitation skill-level category that denotes how hard it is to successfully exploit the vulnerability. Then, based on historical attack data, we can create a statistical model of the exploitation skill of the attacker community using a Bayesian approach. Last, the final security risk is computed by marginalizing over the exploitation skill.

For our model, we follow the terminology of the multi-prerequisite attack graph (MP attack graph) presented in [13], but we introduce our own notation. The MP attack graph is fast to create, easy to understand and easy to work with; it uses very few model elements and have been shown to scale to large networks [13]. However, we are not bound to any specific attack graph as long as we can model the flow of data.

Notice however, that our model subsumes the original model such that our model can instantiate an equivalent recommendation algorithm for patching as in [13]. By removing all data flows, assigning all probabilities of successful attacks to the constant value 1.0 and putting data sets with a uniform loss value of 1 at each node, we can reproduce the original recommendation algorithm. By

assigning different loss values to the different data sets, we also can reproduce the weighted version of the original algorithm. In addition, in contrast to the original model, our model makes it possible to apply partial patches that instead of removing a vulnerability, increase the required exploitation skill-level.

The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the impact of vulnerabilities to an IT-system [15]. We take advantage of the expert knowledge gathered through CVSS by using the impact elements from CVSS version 2.

A simple network from [13] that is shown in Fig. 1(a) illustrates what sort of problem we want to address with our approach. In this simple example, attacks start at host A with root access. All other hosts, but the firewall (FW), have a single vulnerability instance, e.g. v_B , that can be exploited remotely via one single open port. The firewall will accept that hosts C and D communicate with host E and will deny all other communication. In Fig. 1(b), we show the MP attack graph where the triangles indicate vulnerability nodes, circles indicate attack states and rectangles indicate what nodes can be reached from an attack state. The problem we are addressing is how to compare the security of different networks with respect to their vulnerabilities.

The rest of the paper is organized as follows. Sect. 2 introduces our formalization of the network model and the attack graph from [13]. Sect. 3 describes the network impact model. Sect. 4 presents the model for computing the probability that an attack will successfully exploit a vulnerability. Sect. 5 describes how to compute the security risk given a historical record of attack data. Sect. 6 applies our framework to the simple network example described above. Sect. 7 compares related work with the proposed framework.

2 Introducing the MP Attack Graph Model

Our approach begins with constructing a model of the network containing elements such as a network topology, and traffic rules. Then, we define the MP attack graph. In the next sections, we follow the terminology of the NetSPA tool presented in [13], but we introduce our own notation.

Basic Network Model. A *network topology* consists of a set of *hosts* N and a set of *links* E . Each host $h \in N$ has a set of *interfaces* $i(h) \subseteq I$ where I is the set of all interfaces in the network. Each link $l \in E$ connects a set of interfaces $i(l) \subseteq I$. Each interface $i \in i(h)$ has a set of *ports* $p(h, i)$.

A traffic rule allows source and destination software instances to communicate via two interfaces of a host. Each host $h \in N$ has a set of *traffic rules* $r(h)$. By setting the second interface in a traffic rule equal to the first, we can allow traffic to only flow from or to the host of the rule.

MP Attack Graph. The MP attack graph consists of three types of nodes: *prerequisites*, *attack states* and *vulnerability instances*.

A *prerequisite* is the means needed to gain access to a vulnerability instance. We use Q to denote the set of all prerequisites in a network. For each prerequisite $q \in Q$, $v(q)$ denotes the set of (reachable) vulnerability instances of q .

An *attack state* a consists of a host $h \in N$ and an attack level $o \in \{\text{root}, \text{user}, \text{dos}, \text{other}\}$. We use A to denote the set of all attack states. For each $a \in A$, an attacker will, by obtaining attack state a , also obtain a set of prerequisites $q(a) \subseteq Q$ available at a .

A *vulnerability instance* is any means that an attacker can use to gain access to a system. We let V denote the set of all vulnerability instances. Each software instance $s \in S$ has a set of vulnerability instances $v(s) \subseteq V$. By successfully exploiting a vulnerability instance $v \in V$, an attacker will obtain a single attack state $a(v) \in A$.

3 Modeling the Impact to a Network Using Data Flows

After defining the basic network model and the attack graph, we define the data model and the impact model. These are our own contributions, except for some borrowed notions from [15]. A more elaborate description of the network impact model can be found in [16].

3.1 Defining the Data Model

A *data set instance* is a copy of any set of data that we protect with respect to the three security aspects – confidentiality, integrity and availability – that we denote c , i and a . D denotes the set of all data set instances.

A *data set instance location* consists of a data set instance $d \in D$ and a location identifier with a host $h \in N$, an interface $i \in i(h)$ of host h and a port $p \in p(h, i)$ of interface i .

All data set instances are assumed to be processed by *software instances*. S denote the set of all software instances. For each host $h \in N$, $s(h) \subseteq S$ denotes all software instances at h . For each software instance $s \in s(h)$, $store(s)$ denotes all data set instances stored at s and for each interface $i \in i(h)$, $inputs(s, i)$ and $outputs(s, i)$ denotes the set of data set locations from where s retrieves and produces data set instances via interface i respectively.

We let $depends(s, d)$ denote the set of data set instances used by s to produce data set d . Then, either, for each data set instance $d \in depends(s, d')$, $d \in store(s)$ or there is an interface $i \in i(h)$ such that d has a data set instance location in $inputs(s, i)$.

A *data flow* for data set instance $d \in D$ from a producer software instance $s_0 \in S$ to a retriever software instance $s_n \in S$ is a sequence of hosts $(h_0, h_1, h_2, \dots, h_{n-1}, h_n)$ in N where: (a) $n > 0$, (b) $outputs(s_0, i_0)$ and $inputs(s_n, i_n)$ contain the same data set instance location with host h_0 , interface i_0 , port $p_0 \in p(h_0, i_0)$ and data set instance d , (c) there exists links with interfaces connecting all hosts in the sequence, and (d) there exists a traffic rule at each host allowing traffic from source host h_0 to destination host h_n using as source and destination the location identifiers of the producer and the retriever softwares respectively.

In addition, a data flow f is *active* if the data set instance of f is depending on other data set instances through the *depends* relation and then each of these

data sets are either stored at the producer or retrieved such that there must exist an active data flow f' for the retrieved data set instance with the producer of f as the receiver of f' .

3.2 Defining the Impact Model

As a means to define the local impact, we define that a data set instance $d \in D$ is *accessible* to a software instance $s \in S$ if either $d \in store(s)$ (d is stored), there exists an active data flow for d with s as receiver (d is retrieved) or there exists an interface i such that d has a data set instance location in $outputs(s, i)$ while for each $d' \in depends(s, d)$, either $d' \in store(s, d)$ or there exists an active data flow for d' with s as receiver (d is produced). Similarly, a data set instance $d \in D$ is *accessible* to a host $h \in N$ if there exists a software instance $s' \in s(h)$ such that d is accessible to s' or there exists an active data flow for d with a sequence of hosts containing h .

Since we want to model the impact of a successful attack to the three security aspects, confidentiality, integrity and availability, we define that for each vulnerability instances $v \in V$ and for each security aspect $e \in \{c, i, a\}$, $impact(v, e) \in \{None, Partial, Complete\}$ denotes the local impact that vulnerability instance v has on security aspect e at a host h . We borrow the values *None*, *Partial*, *Complete* for the three security aspects from CVSSv2 [15].

For our purpose, we interpret the impact values of CVSSv2 such that for each security aspect $e \in \{c, i, a\}$, an attacker will *locally violate* the security aspect e of, in the case of impact value: (a) *None*, no data set instances, (b) *Partial*, all data set instances *accessible* to the software instance with an exploited vulnerability instance, and (c) *Complete*, all data set instances *accessible* to the host of the software instance with the exploited vulnerability instance. Then, we define that for each data instance $d \in D$, for each security aspect $e \in \{c, i, a\}$, for each subset of exploited vulnerability instances $V_{exploit} \subseteq V$ and for each host $h \in N$, $locallyViolated(e, d, h, V_{exp})$ denotes that d was locally violated by an attacker at host h with respect to e given $V_{Exploit}$.

Now, we define predicates that infer violation of data set instances at the network level:

Loss of confidentiality. For each data set instance $d \in D$, for each host $h \in N$ and for each subset of vulnerability instances $V_{exp} \subseteq V$, $lossOfConf(d, h, V_{exp})$ holds true if $locallyViolated(c, d, h, V_{exp})$ holds true.

Loss of integrity. For each data set instance $d \in D$, for each host $h \in N$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfInteg(d, h, V_{exp})$ holds true if either (a) $locallyViolated(i, d, h, V_{exp})$ holds true, or (b) there exists $s \in S$ such that $d' \in depends(s, d)$ and there exists an active data flow f for d' where the receiver is s and there exists a host h' in the sequence of hosts of f where $h' \neq h$ and $lossOfInteg(d', h', V_{exp})$ holds true.

Loss of availability. For each data set instance $d \in D$, for each host $h \in N$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfAvail(d, h, V_{exp})$ holds true if either (a) $locallyViolated(a, d, h, V_{exp})$ holds true, or (b)

there exists $s \in S$ such that $d' \in \text{depends}(s, d)$ and for each active data flow f for d' where the receiver is s there exists a host h' in the sequence of hosts of f where $h' \neq h$ and $\text{lossOfAvail}(d', h', V_{exp})$ holds true.

Generic loss of security. For each security aspect $e \in \{c, i, a\}$, for each data set instance $d \in D$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $\text{lossOfSecurity}(e, d, V_{exp})$ holds true if there exists $h \in N$ such that either $e = c$ and $\text{lossOfConf}(d, h, V_{exp})$ or $e = i$ and $\text{lossOfInteg}(d, h, V_{exp})$ or $e = a$ and $\text{lossOfAvail}(d, h, V_{exp})$ holds true.

Lastly, by assigning a *loss value*, we can estimate a cost of exploiting a vulnerability instance with a certain loss of security. For each data set instance $d \in D$, and for each security aspect $e \in \{c, i, a\}$, $\text{cost}(d, e)$ denotes the loss value of data set d with respect to e .

4 Modeling the Probability of a Successful Attack

In [13], the authors assume a worst case scenario for a successful attack: a single attacker, starting at an initial attack state, will be able to exploit every vulnerability instance in the MP attack graph. However, it is not a reasonable assumption, since attackers might have different exploitation skills and vulnerability instances might have different required exploitation skills. We propose a different scenario where an attacker will be able to *try* to exploit all vulnerability instances it has gained access to, but that the probability of successfully exploit a vulnerability instance depends on the *required exploitation skill-level* of the vulnerability instance and the *exploitation skill* of the attacker. Notice that we view each attacker as an instantiation of the wider attacker community, and thus, we do not model each attacker individually, but as a group.

For convenience, we have chosen to use four different exploitation skill-levels that we define in terms of a required exploitation skill. We map each exploitation skill-level into a probability distribution over the required exploitation skill. Similarly, we have chosen the exploitation skill to be in the arbitrary range [0,1].

In addition, we assume that an expert has assigned a exploitation skill-level to each vulnerability instance, for instance, by analyzing a freely available database such as NVD [17].

Required Exploitation Skill-Level. First we define the required exploitation skill-level. For each vulnerability instance $v \in V$, $z(v) \in \{Low, MediumLow, MediumHigh, High\}$ denotes the exploitation skill-level required by an attacker to successfully exploit v . To make formulas shorter, we sometimes use the following notation: $z_0 = Low$, $z_1 = MediumLow$, $z_2 = MediumHigh$ and $z_3 = High$. Thus, instead of typing $z(v) = Low$, we type $z(v) = z_0$ and so forth.

Successful Exploitation. Then, we define the successful exploitation probability. For each vulnerability instance $v \in V$, $k \in [0, 1]$ denotes the actual exploitation skill of an attacker, $k_v \in [0, 1]$ denotes the required exploitation to successfully exploit v , $p(k_v | m_z, \sigma_z)$ denotes the probability density

function over k_v (where k_v is normally distributed with parameters m_z and σ_z and $z = z(v)$), and $\Phi(k, z)$ denotes the probability that an attacker with exploitation skill k will succeed to exploit v :

$$\forall i \in \{0, 1, 2, 3\}, m_{z_i} = \frac{i}{4} + \frac{1}{8} \text{ and } \sigma_{z_i} = \frac{1}{5}, \quad (1)$$

$$\Phi(k, z) \propto \int_k^1 p(k_v | m_z, \sigma_z) dk_v = \text{erf} \left(\frac{k - m_z}{\sigma_z \sqrt{2}} \right) - \text{erf} \left(\frac{-m_z}{\sigma_z \sqrt{2}} \right) \quad (2)$$

where erf is the error function [18].

5 Computing the Risk

In order to compute the risk to a network, without relying too much on expert knowledge, we use Bayesian statistical inference over a historical record of attack data.

To gather historical attack data for whether vulnerability instances were successfully exploited or not, we propose using a high-interactive honeypot. A honeypot is a controlled computer created to be vulnerable to attacks and therefore, any occurring traffic is suspicious [19]. Hence, it is easy to detect attack attempts. If the nature of a set of vulnerability instances in the honeypot is known, it should also be possible to verify whether they were successfully exploited [20]. Of course, it is also possible to add other compromises investigated by the operator of the network.

Notice though, that to keep the probability estimations up-to-date, we must limit how old attack data we take into account. For instance, it might be realistic to only use the last six months of data, since older data might make the risk value obsolete.

Recall from Sect. 1 that risk is a function of an impact value and the probability of a successful attack. Therefore, we define the total risk to a network from attacks starting at $a_0 \in A$ in attack graph \mathcal{A} during next s time slots as the expected loss over all data set instances:

$$\text{risk}(a_0, s, X, \mathcal{A}) = \sum_{d \in D, e \in \{c, i, a\}} \text{cost}(d, e) \cdot P_e(d | a_0, s, X, \mathcal{A}) \quad (3)$$

where $P_e(d | a_0, s, X, \mathcal{A})$ is the *probability of at least one successful attack* on data d given the *historical attack data* X , a_0 and s .

Historical Attack Data. First, we define the historical record of attack data X over a time period T where X contains $n(X)$ number of attacks and T contains $n(T)$ number of time slots. For each attack data instance $x \in X$, $z(x) \in \{z_0, z_1, z_2, z_3\}$ denotes the required exploitation skill-level of the vulnerability instance during the attack, and $e(x) \in \{0, 1\}$ denotes whether the attack was successful ($e(x) = 1$) or not ($e(x) = 0$).

Posterior Distribution. Second, we derive the posterior distribution for the exploitation skill k and a parameter λ , which governs the number of expected attack instances, using Bayesian inference. The posterior distribution given X is:

$$P(k, \lambda|X) \propto P(X|k, \lambda) \cdot P(k) \cdot P(\lambda) \tag{4}$$

where $P(X|k, \lambda)$ denotes the probability that X was generated by an attacker community with exploitation skill k and parameter λ , while $P(k)$ and $P(\lambda)$ denotes the prior distributions for k and λ , respectively. We let $P(k) \propto 1$ (uniform distribution) and $P(\lambda) = \textit{Gamma}(a, b)$ (Gamma distribution). The prior distributions describe what we know about the parameters prior to having the real data. Assuming independence between k and λ , we have:

$$P(X|k, \lambda) = P(X|k) \cdot P(n(X)|\lambda) \tag{5}$$

where $P(X|k)$ is the probability of X given k and $P(n(X)|\lambda)$ is the probability of $n(X)$ number of attack instances during time period T . For $P(X|k)$ we have:

$$P(X|k) \propto \prod_{x \in X} P(x|k) \tag{6}$$

where for each attack data instance $x \in X$, $P(x|k)$ denotes the probability distribution over x given exploitation skill $k \in [0, 1]$:

$$P(x|k) = (1 - \Phi(k, z))^{(1-e(x))} \cdot \Phi(k, z)^{e(x)} \text{ where } z = z(x) \tag{7}$$

such that

$$P(X|k) \propto \prod_{i=0}^3 (1 - \Phi(k, z_i))^{e_{-z_i}} \cdot \Phi(k, z_i)^{e_{z_i}} \tag{8}$$

where e_{-z_i} is the total number of unsuccessful attacks and e_{z_i} is the number of successful attacks on vulnerability instances with required skill-level z_i . Then, for $P(n(X)|\lambda)$ we assume a Poisson distribution with mean value $n(T) \cdot \lambda$ such that:

$$P(n(X)|\lambda) = \frac{(n(T)\lambda)^{n(X)} e^{-n(T)\lambda}}{n(X)!} \tag{9}$$

Probability of Success. Last, we derive the probability of at least one successful attack by marginalization. In case of attack graph \mathcal{A} and attacks starting in attack state $a_0 \in A$, for each data set instances $d \in D$ and for each security aspects $e \in \{c, i, a\}$, $P_e(d|a_0, \mathcal{A}, k)$ denotes the probability that an attacker from an attacker community with exploitation skill $k \in [0, 1]$ will successfully exploit vulnerability instances in \mathcal{A} such that d will be compromised with respect to e , and $P_e(d^n|a_0, \mathcal{A}, k)$ denotes the probability that at least one of n attack attempts succeeds:

$$P_e(d^n|a_0, \mathcal{A}, k) = 1 - (1 - P_e(d|a_0, \mathcal{A}, k))^n \tag{10}$$

Then, we can compute the probability of at least one successful attack on d with respect to e , given X and within next s time slots, as:

$$\begin{aligned}
 P_e(d|a_0, s, X, \mathcal{A}) &\propto \\
 &\int_0^1 \int_0^\infty \sum_{n=0}^\infty P_e(d^n|a_0, \mathcal{A}, k) P(n(s)|\lambda) P(X|k, \lambda) p(k) p(\lambda) d\lambda dk = \\
 &C - \int_0^1 \frac{1}{\left(1 + \frac{sP_e(d|a_0, \mathcal{A}, k)}{b+n(T)}\right)^{a+n(X)}} \prod_{i=0}^3 (1 - \Phi(k, z_i))^{e-z_i} \Phi(k, z_i)^{e-z_i} dk
 \end{aligned} \tag{11}$$

where $n(s)$ is the number of attacks during next s time slots, for which we use a Poisson distribution with mean $s \cdot \lambda$, and C is a constant.

We compute the marginal distribution in (11) using a Monte-Carlo simulation, since computing $P_e(d|a_0, \mathcal{A}, k)$ for the generic problem is NP-complete for reasonably complex networks [21].

The Monte-Carlo algorithm (Appendix A) is a variant of the algorithm for probabilistic networks with random links presented in [21]. In our case, instead of links, vulnerabilities can be randomly exploited or not, but where the exploitation of a vulnerability might require that some other vulnerabilities must already have been exploited. Our algorithm starts from the initial attack state, and then randomly draws an exploitation skill k . Thereafter, we follow the MP attack graph by randomly exploiting all vulnerability instances reachable from the prerequisites of the initial attack state. Next, we repeat the last step for all reachable attack states of the exploited vulnerability instances, until we have tried to exploit all vulnerability instances gained access to through prerequisites. Then, from a large number of repetition of the previous steps, while keep starting the attacks from the initial attack state, we can use our predicates to check for loss of security for any data set instance, and finally, estimate probabilities of successful attacks.

6 Modeling and Analyzing the Simple Network Example

In Table 1-2, we model the simple network example from Sect. 1. The network in Fig. 1(a) is modeled in Table 1 and the MP attack graph shown in Fig. 1(b) is modeled in Table 2. Table 3 and Table 4 shows a data model as well as an impact model respectively. The data model models that data set d_E is located at host E , d_F is located at F and that $d_{E'}$ is dependent of d_E for its existence and that $d_{E'}$ flows from E to C . We let all vulnerabilities have the same required exploitation skill-level.

In Fig. 6, each curve shows the expected risk during the next time slot ($s = 1$) for one of the required exploitation skill-levels. The expected risk is shown on the y-axis (maximum risk is 10.5) and the time on the x-axis. At time point zero, there are no attacks, but because of our prior distributions, we can infer a risk anyway. Thereafter, an attack attempt is recorded at each time point as

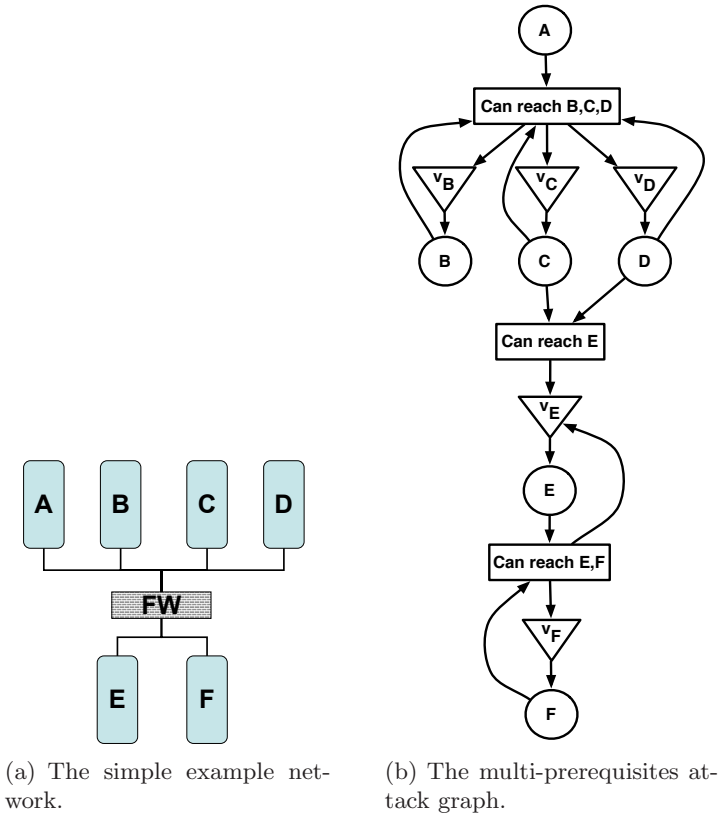


Fig. 1. The simple example from [13]

shown in Table 5. A time slot is 15 time points and thus, the number of time slots $n(T)$ is $1, 2, \dots, 7$ depending on at what time the risk is computed.

Our prior distribution $p(k) \propto 1$ implies the cautious, prior belief that the exploitation skill is uniformly distributed in the attacker community, while for the prior distribution of λ we set $a = 1$ and $b = 1$ in (11) such that $p(\lambda) = e^{-\lambda}$.

As we might have expected, each curve looks different given a different required exploitation skill-level. Not surprisingly, for all curves, the risk decreases given evidence of unsuccessful attacks, while the risk increases when given evidence of successful attacks. Reasonable enough, the Low-curve is most sensitive to all types of attacks, while attacks at a lower skill-level have lesser impact on a risk curve with a higher required skill-level.

We can now compare our model with the recommendation algorithm for patches in [13]. Table 6 shows the impact estimation when the probability of success is 1.0 for all vulnerabilities. We have compared three different data models and five different patch choices. In [13], patches of set of vulnerabilities are compared. For this simple example, we only patch a single vulnerability instance. Column 2 shows the impact estimation of the models in Table 3 and 4. Column 3

Table 1. Network model

$$\begin{aligned}
N &= \{A, B, C, D, FW, E, F\} \\
E &= \{l_1, l_2\} \\
i(FW) &= \{i_{FW1}, i_{FW2}\} \\
\forall n \in N_0 : i(n) &= \{i_n\} \\
i(l_1) &= \{i_A, i_B, i_C, i_D, i_{FW1}\} \\
i(l_2) &= \{i_{FW2}, i_E, i_F\} \\
r(FW) &= \{\langle i_{FW1}, i_{FW2}, [C, i_C, 0], \\
&[E, i_E, 0] \rangle, \langle i_{FW1}, i_{FW2}, [D, i_D, 0], \\
&[E, i_E, 0] \rangle\} \\
r(C) &= \{\langle i_C, i_C, [C, i_C, 0], [E, i_E, 0] \rangle\} \\
r(D) &= \{\langle i_D, i_D, [D, i_D, 0], [E, i_E, 0] \rangle\} \\
r(E) &= \{\langle i_E, i_E, [C, i_C, 0], [E, i_E, 0] \rangle, \\
&\langle i_E, i_E, [D, i_D, 0], [E, i_E, 0] \rangle\}
\end{aligned}$$

Table 3. Data model

$$\begin{aligned}
S &= \{s_B, s_C, s_D, s_E, s_F\} \\
\forall h \in N - \{A, FW\} : s(h) &= \{s_h\} \\
D &= \{d_E, d_{E'}, d_F\} \\
store(s_F) &= \{d_F\} \\
store(s_E) &= \{d_E\} \\
dependencies(s_E, d_{E'}) &= \{d_E\} \\
outputs(s_E, i_E) &= \{\langle d_{E'}, E, i_E, 0 \rangle\} \\
inputs(s_C, i_C) &= \{\langle d_{E'}, E, i_E, 0 \rangle\}
\end{aligned}$$

Table 2. MP attack graph

$$\begin{aligned}
A &= \{a_A, a_B, a_C, a_D, a_E, a_F\} \\
Q &= \{q_{BCD}, q_E, q_{EF}\} \\
v(q_{BCD}) &= \{v_B, v_C, v_D\} \\
v(q_E) &= \{v_E\} \\
v(q_{EF}) &= \{v_E, v_F\} \\
q(a_A) &= q(a_B) = \{q_{BCD}\} \\
q(a_C) &= \{q_{BCD}, q_E\} \\
q(a_D) &= \{q_{BCD}, q_E\} \\
q(a_E) &= \{q_{EF}\}, q(a_F) = \{q_{EF}\} \\
a(v_B) &= a_B, a(v_C) = a_C \\
a(v_D) &= a_D, a(v_E) = a_E, a(v_F) = a_F
\end{aligned}$$

Table 4. Impact Model

$$\begin{aligned}
V &= \{v_B, v_C, v_D, v_E, v_F\} \\
\forall h \in N - \{A, FW\} : v(s_h) &= \{v_h\} \\
\forall h \in N - \{A, FW\}, e \in \{c, i, a\} : \\
& \quad impact(v_h, e) = Complete \\
\forall e \in \{c, i, a\} : loss(d_E, e) &= 1 \\
\forall e \in \{c, i, a\} : loss(d_{E'}, e) &= 0.5 \\
\forall e \in \{c, i, a\} : loss(d_F, e) &= 2
\end{aligned}$$

shows the values when we remove the data flow, in which case $d_{E'}$ cannot be reached anymore because v_E is patched. Column 4 has the values for the case when each host $B - F$ has a data set instance with loss value 1. Notice that patching v_E is the best choice in all cases.

The column 3 and 4 of Table 6 can easily be replicated by the algorithm in 13, but not column 2. Column 3 is replicated by letting E have weight 4.5 and F weight 6 and column 4 by letting $B - F$ have weight 1. However, consider the second column, if we use the same weights as for the third column, we would have the result in the first column in Table 7, where the value of patching v_E differs from that in column 2 in Table 6. To fix this, we can assign weight 3 to E , 1.5 to C and 6 to F , but then we get the result in column 2 of Table 7, where instead the value of patching v_C differs from that in column 2 in Table 6. Consequently, the importance of a host depends on the attack graph in case of the original algorithm, while this is not the case for our improved model. The problem is that in order to compromise data set instance $d_{E'}$, we only have to compromise either C or E . Thus compromising one of them is enough. However, this is not possible to express using the simple weight assignment.

In Fig. 6, we show the estimated risk using our statistical model, corresponding to column 1 in Table 6. We let all vulnerabilities have required skill-level *Low* while using the attacks from Table 5. As can be seen, the risk curves of our

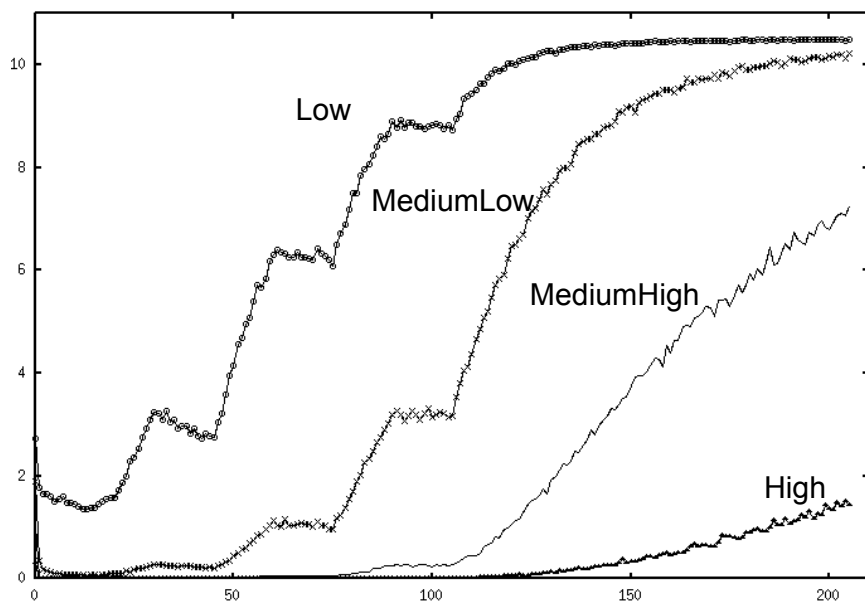


Fig. 2. Risk as function of time according to required exploitation skill-level

Table 5. Attack data; a time slot is 15 time points

Time	Skill-level	Success.	#attacks
0	none	-	0
1 - 15	Low	No	15
16 - 30	Low	Yes	30
31 - 45	MedLow	No	45
46 - 60	MedLow	Yes	60
61 - 75	MedHigh	No	75
76 - 90	MedHigh	Yes	90
90 - 105	High	No	105
106 - 205	High	Yes	205

Table 6. Impact for three different data models

Patch	Unmod	No Flow	Uniform
<i>none</i>	10.5	10.5	5
<i>v_B</i>	10.5	10.5	4
<i>v_C</i>	10.5	10.5	4
<i>v_D</i>	10.5	10.5	4
<i>v_E</i>	1.5	0	3
<i>v_F</i>	4.5	4.5	4

Table 7. Impact for original model

Original 1	Original 2
10.5	10.5
10.5	10.5
10.5	9
10.5	10.5
0	1.5
4.5	4.5

algorithm converge into the same values, given enough attack evidence, as in column 1 in Table 6. Thus similarly, by removing vulnerability instance v_E we would have the least risk regardless of attack data. Thus, v_e would be our best recommendation. The next best would be v_F . However, if we would have made the recommendation before time point 45, we would instead have recommended vulnerability instance v_C as the next best recommendation. Notice also that at time step 100, removing v_D or v_C would be better than removing v_B or doing nothing. Accordingly, in contrast to the original algorithm that produces no difference in risk/impact over time, our algorithm makes it possible to make fine grained prioritization of patches, using not only expert knowledge as input, but also monitor data.

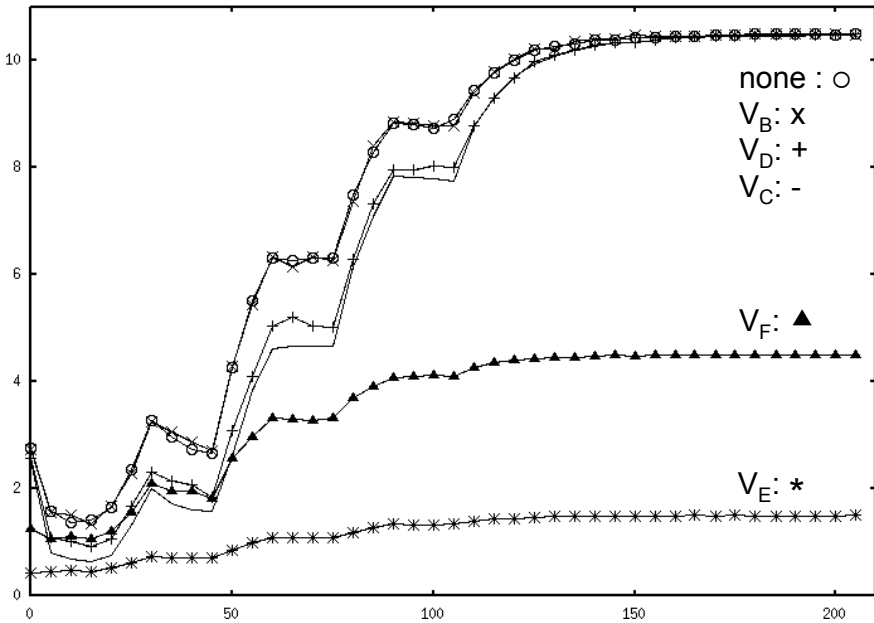


Fig. 3. The risk curves for patching one vulnerability

7 Related work

A set of papers [3,4,5,6,7] define metrics to measure the security of networks based primarily on different ways of weighing paths in an attack graph. Other approaches – like our work – use the risk as a measure of the security of a network where the cost or loss of an intrusion is used as the basis for the metric [8,9,10,11].

An early work in this area is presented in [3,4]. As security metrics the authors use the *time* from an attack starts until it succeeds and the *effort* required by the attacker to succeed. They use a Markov model over a privilege graph to compute

the two metrics, where they assume exponential distributions for the required time and effort. Their metrics are complementary to our work in that they consider the *cost* for an attacker, while we consider the loss to the system owners.

A work inspired by reliability analysis is presented in [5,6]. The authors use a model checker to construct an attack graph from a detailed model of the system. As a security metric they propose the probability that an attacker will end up in a unsafe system state. They estimate the probabilities by the use of a Markov Decision Process (MDP) [22]. In contrast to our work, they don't consider the different skill required to use an exploit. In addition, they do not suggest how to come up with estimations of the transition probabilities of the MPD.

In [7] the authors use the PageRank algorithm of Google to measure the security of a network from an attack graph. The result is an ergodic Markov chain that converges into a probability distribution over the attack states. In contrast to our work, they do not consider the value of protected assets or dependencies between attack states; neither do they suggest how to use historical data to derive transition probabilities.

In [8], the authors present a framework called RheoStat that uses a risk-based analysis to select a response. Due to that RheoStat's likelihood metric is not probability based, RheoStat can only estimate the risk conditioned on the current intrusion alerts, while our work also can estimate the risk given the intrusion activities over time. Thus, our work can be used to compare different network configurations.

The work in [9] uses a Hidden Markov Model (HMM) to estimate the probability that nodes in a network are in malicious states given observed intrusion alerts from an IDS. In contrast to our work, the authors only assign costs of each host being in a malicious state. The impact is estimated either as the total expected cost for all hosts or the average expected cost per host. They neither consider the value of protected assets nor dependencies between host.

Another approach to real-time risk analysis uses Hidden Markov Models as input to a Fuzzy inference system [10]. A set of Fuzzy rules are used to derive three linguistic variables: intrusion frequency, probability of threat success and severity. The HMMs provide an estimation of the intrusion frequency of different attack types, while the other input values come from a distributed intrusion detection system and from a traffic rate monitor. Then another set of Fuzzy rules infer the final risk assessment from the three linguistic variables. In [11], the same authors present an extension where the Fuzzy rules are optimized using a neural network that learn from given training examples. The papers are brief on how the system works and how it was tested, hence a comparison with our approach is not easy to do.

8 Summary and Concluding Remarks

We have presented a framework for assessing the security risk to a network using data flows over an attack graph and a Bayesian model of the exploitation skill of the attacker community, given a record of attack data. By modeling the flow

of data, we take into account dependencies between different hosts and different data flows. In addition, we are able to update the risk computation as soon as we receive more attack data.

We also propose using a honeypot to collect historical attack data. However, in order to model the exploitation skill, each vulnerability instance must be classified by an expert into a required exploitation skill-level. This can turn into a problem since a vulnerability that once was hard to exploit might suddenly become easy when somebody creates a downloadable exploitation code. Thus, some monitoring system for updating the information about vulnerabilities is needed as well as a support system for classifying old and new vulnerabilities given new information.

Acknowledgements. This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab Systems, TeliaSonera and T2Data.

References

1. Nessus: The network vulnerability scanner (April 2009), <http://www.nessus.org>
2. Lippman, R., Ingols, K.: An annotated review of past papers on attack graphs. Project Report PR-IA-1, MIT Lincoln laboratory (March 2005)
3. Dacier, M., Deswarte, Y., Kaaniche, M.: Quantitative assessment of operational security: Models and tools (1996)
4. Ortalo, R., Deswarte, Y.: Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering* 25, 633–650 (1999)
5. Jha, S., Sheyner, O., Wing, J.M.: Minimization and reliability analyses of attack graphs. Technical Report CMU-CS-02-109, School of Computer Science, Carnegie Mellon University (2002)
6. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: *IEEE Symposium on Security and Privacy*, pp. 273–284 (2002)
7. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: Zamboni, D., Krügel, C. (eds.) *RAID 2006*. LNCS, vol. 4219, pp. 127–144. Springer, Heidelberg (2006)
8. Gehani, A., Kedem, G.: RheoStat: Real-time risk management. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) *RAID 2004*. LNCS, vol. 3224, pp. 296–314. Springer, Heidelberg (2004)
9. Årnes, A., Valeur, F., Vigna, G., Kemmerer, R.A.: Using hidden markov models to evaluate the risks of intrusions - system architecture and model validation. In: Zamboni, D., Krügel, C. (eds.) *RAID 2006*. LNCS, vol. 4219, pp. 145–164. Springer, Heidelberg (2006)
10. Haslum, K., Abraham, A., Knapskog, S.: Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In: *Third International Symposium on Information Assurance and Security*. IEEE Computer Society Press, Los Alamitos (2007)
11. Haslum, K., Abraham, A., Knapskog, S.J.: Hinfra: Hierarchical neuro-fuzzy learning for online risk assessment. In: *Asia International Conference on Modelling and Simulation*, pp. 631–636 (2008)

12. Stoneburner, G., Goguen, A., Feringa, A.: Risk management guide for information technology systems. Technical Report NIST SP 800-30, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology (2002)
13. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: Computer Security Applications Conference, pp. 121–130 (2006)
14. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis, 2nd edn. Chapman & Hall/CRC, Boca Raton (July 2003)
15. CVSS: Common vulnerability scoring system (April 2009), <http://www.first.org/cvss/>
16. Olsson, T.: Impact estimation using data flows over attack graphs. In: Proceedings of the 14th Nordic Conference on Secure IT Systems, NordSec (2009)
17. NVD: National vulnerability database (April 2009), <http://nvd.nist.gov/>
18. Abramowitz, M., Stegun, I.A. (eds.): Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Dover, New York (1972)
19. McGrew, R., Vaughn, R.B.: Experiences with honeypot systems: Development, deployment, and analysis. In: Proceedings of the 39th Hawaii International Conference on System Sciences (2006)
20. Alata, E., Nicomette, V., Kaâniche, M., Dacier, M., Herrb, M.: Lessons learned from the deployment of a high-interaction honeypot. In: EDCS 2006, 6th European Dependable Computing Conference (October 2006)
21. Marzot, G.S.: Netstat: A probabilistic network connectivity analysis tool. Technical Report A838262, MITRE CORP BEDFORD MA (1993)
22. Filar, J.A., Vrieze, K.: Competitive Markov Decision Processes. Springer, Heidelberg (1996)

A Monte-Carlo Algorithm

Input: a_0 - attack state, d - data set instance, s - time slots, e - security aspect,
 X - attack data

Output: $prob$ - the estimated probability of loss of security.

$N \leftarrow 100000$: Number of sampled values.

$m_e \leftarrow 0$: mean value of the probability of loss of security .

$x \leftarrow 0$: a weighted value, indicating whether d was compromised.

$m_{P(X)} \leftarrow 0$: estimated mean value for probability of the *Data*.

$s_{P(X)} \leftarrow 0$: estimated standard deviation of probability of the *Data*.

for $i = 0; i < N; i = i + 1$ **do**

$k \leftarrow$ sample from $p(k)$;

$StatesCurrent \leftarrow \{a_0\}$, $VulnsCurrent \leftarrow \emptyset$, $StatesUsed \leftarrow \emptyset$;

$VulnsUsed \leftarrow \emptyset$, $PrerequisitesUsed \leftarrow \emptyset$, $VulnsExploited \leftarrow \emptyset$;

 Collect samples of successfully exploitations:

while $StatesCurrent \neq \emptyset$ **do**

foreach $a \in StatesCurrent$ **do**

foreach $q \in q(a)$ and $q \notin PrerequisitesUsed$ **do**

foreach $v \in v(q)$ and $v \notin VulnsUsed$ **do**

$VulnsCurrent \leftarrow VulnsCurrent \cup \{v\}$;

$VulnsUsed \leftarrow VulnsUsed \cup v(q)$;

$PrerequisitesUsed \leftarrow PrerequisitesUsed \cup q(a)$;

$StatesUsed \leftarrow StatesUsed \cup StatesCurrent$, $StatesCurrent \leftarrow \emptyset$;

foreach $v \in VulnsCurrent$ **do**

if $rand(0, 1) > 1 - \Phi(k, m_{z(v)}, \sigma_{z(v)})$ and $a(v) \notin StatesUsed$ **then**

$StatesCurrent \leftarrow StatesCurrent \cup \{a(v)\}$;

$VulnsExploited \leftarrow VulnsExploited \cup \{v\}$;

$VulnsCurrent \leftarrow \emptyset$;

 Estimate current mean value:

if $lossOfSecurity(e, d, VulnsExploited)$ holds true **then**

$x \leftarrow \left(\frac{n(T)+b}{n(T)+s+b} \right)^{n(X)+a} \cdot p(X|k)$ **else** $x \leftarrow P(X|k)$;

if $i = 0$ **then**

$m'_e \leftarrow x$, $s'_e \leftarrow 0$, $m'_{P(X)} \leftarrow P(X|k)$, $s'_{P(X)} \leftarrow 0$: Init old values

else

$m_e \leftarrow m'_e + (x - m'_e)/(i + 2)$,

$m_{P(X)} \leftarrow m'_{P(X)} + (P(X|k) - m'_{P(X)})/(i + 2)$

$m'_e \leftarrow m_e$, $s'_e \leftarrow s_e$, $m'_{P(X)} \leftarrow m_{P(X)}$, $s'_{P(X)} \leftarrow s_{P(X)}$: Save old

 values

$prob \leftarrow 1 - \frac{m_e}{m_{P(X)}}$;

Using the (Open) Solaris Service Management Facility as a Building Block for System Security

Christoph Schuba

Sun Microsystems, Inc.
17 Network Circle
Menlo Park, CA 94025, USA
Christoph.Schuba@Sun.com

Abstract. This paper presents how the Solaris Service Management Facility (SMF) is used as a fundamental building block to improve system security. The Service Management Facility is a backwards-compatible extension to the traditional way Unix services are managed with the `rc` (run command) utility command scripts.

As an integrated framework for managing services and service instances, the SMF improves service availability through automatic correction of failed services in dependency order. It also serves as a launch pad for unmodified, often third party services to be transparently started under the Solaris privilege process rights management without the need to modify source code. Furthermore, different system profiles can be defined that allow a system to come up with or change at runtime into a predefined set of services. Finally, the SMF and service administration are tightly integrated into the Solaris administrative Role-Based Access Control (RBAC) model, subject to the principle of least privilege with strong audit and full administrator accountability.

1 Introduction

In spite of many years and much progress in research and development of system security technologies, it is still a major challenge to configure and maintain servers in a state as resilient to security exploitation as desirable. UNIX operating systems have traditionally included a set of services: software programs not associated with any interactive user login that listen for and respond to requests to perform certain tasks, such as delivering email, responding to ftp requests, or permitting remote command execution. These traditional services were usually individual applications that executed as a single process that started at boot time and executed continuously while a system was up and running, servicing any requests that were received.

Today, administrators must contend with a collection of services that has grown to such a point that it has exceeded the utility of this original model. Sun has created the Service Management Facility (SMF) to simplify management of these system services. The SMF is a feature of the Solaris Operating System that creates a supported, unified model for services and service management on each Solaris system. It is a core part of the Predictive Self-Healing technology

available in Solaris 10, which provides automatic recovery from software and hardware failures as well as administrative errors.

2 Background

The following subsections give some technical background on the security technologies in the (Open)Solaris operating environment that are tied together through the Service Management Facility. While this section describes the least privilege rights management model, the Role-Based Access Control (RBAC) model, and the Secure by Default stance, the following section explains in detail, how they all come together through the SMF.

While other security-related system services, such as the Solaris Cryptographic Framework ([\[7\]](#)), are managed by SMF, this paper focuses on how the system itself is better protected through SMF rather than through the services SMF manages.

2.1 Solaris Privilege Process Rights Management

The principle of least privilege states that every program and every user of a system should operate using the least set of privileges necessary to complete the job. The Solaris operating environment has two primary mechanisms to implement this principle: fine-grained process rights for executable code and role-based access control for administrators. The following sections explain these two mechanisms in detail.

Solaris Privileges. The Solaris operating system implements a set of privileges that provide fine-grained control over the actions of processes. Traditionally, Unix-based systems have relied on the concept of a specially-identified superuser, called root. This concept of a Unix superuser has been replaced in a backward compatible manner with the ability to grant one or more specific privileges that enable processes to perform otherwise restricted operations. The privilege-based security model is equally applicable to processes running under user id 0 (root) or under any other user id. For root-owned processes, the ability to access and modify critical system resources is restricted by removing privileges from these processes. For user-owned processes privileges are added to explicitly allow them to access such critical resources. The implications for such privilege-aware processes are both, that root processes can run more safely because their powers are limited, and that many processes that formerly required to be root processes can now be executed by regular users by simply giving them the additionally required privileges.

Experience with modifying a large set of Solaris programs to be privilege-aware revealed an interesting fact. Most programs that formerly required to be executed as user root require only very few additional privileges and in many cases require them only once before they can be relinquished. The change to a primarily privilege-based security model in the Solaris operating system gives

developers an opportunity to restrict processes to those privileged operations actually needed instead of having to choose between all privileges (superuser) or no privileges (non-zero UIDs). Additionally, a set of previously unrestricted operations now require a privilege; these privileges are dubbed the "basic" privileges. These are privileges that used to be always available to unprivileged processes. By default, processes still have the basic privileges. The single, all-powerful UID 0 assigned to a root process has been replaced with more than 70⁴ discrete privileges that can be individually assigned to processes using the Service Management Facility (SMF), Role-based Access Control (RBAC), or a command-line program, such as `ppriv(1)` or `pfexec(1)`.

Taken together, all defined privileges with the exception of the basic privileges compose the set of privileges that are traditionally associated with the root user. The basic privileges are those privileges unprivileged processes were accustomed to having. They are also called the basic set and consist of `file_link_any`, `proc_exec`, `proc_fork`, `proc_info`, and `proc_session`. The privilege implementation in Solaris extends the process credential with four privilege sets:

- I, the inheritable set: The privileges inherited on exec.
- P, the permitted set: The maximum set of privileges for the process.
- E, the effective set: The privileges currently in effect.
- L, the limit set: The upper bound of the privileges a process and its offspring can obtain.

The implementation of Solaris privileges empowers application developers to control how privileges are used within their programs. Using a technique called privilege bracketing, developers can write their programs such that they are only running with privileges in the effective set when they are needed by certain system calls. Even more importantly, programs can not only enable or disable their privileges (aka privilege bracketing), but they can also drop any privileges granted to them (assuming they will never be needed) and even relinquish them (so they can no longer be used) when there is no longer a need for the privilege. Just as importantly, programs can also restrict which of their privileges can be passed along to their children (e.g., programs that they execute). In the Solaris operating system many setuid programs (e.g., ping, traceroute, rmformat) and system services (e.g., nfsd, ftpd, mountd) use these techniques.

Solaris RBAC. Role-based Access Control (RBAC) in Solaris is an alternative to the all-or-nothing security model of traditional superuser-based systems. With RBAC ([4](#)), an administrator can assign privileged functions to specific user accounts (or special accounts, called roles). RBAC is in keeping with the security principle of least privilege by allowing organizations to selectively grant privileges to users or roles based upon their unique needs and requirements.

* The set of Solaris privileges is expanding over time. As of build 112, April 6, 2009, the OpenSolaris operating systems makes 75 discrete privileges available, only 5 of which represent user space privileges for backwards compatibility for non privilege-aware software.

In general, organizations are strongly encouraged to use Solaris RBAC to restrict access to privileged operations rather than granting users complete access to the backwardly compatible root account. Solaris RBAC was introduced in the Solaris 8 operating system, having come from Trusted Solaris, and has been enhanced and expanded with each new release of Solaris. Solaris RBAC functionality contains several discrete elements that can be used individually or together including authorizations, privileges, rights profiles and role designations.

2.2 Secure by Default

Traditionally, Unix systems have provided a large number of network services by default. This open approach is convenient, but it also makes it easy for remote attackers to exploit any vulnerabilities that may exist in the software providing the network services. The Solaris Secure by Default stance reduces this attack surface by disabling as many network services as possible while still leaving a useful system.

Secure by Default changes the default configuration of Solaris so that `ssh` is the only network-listening service. Other network services are either disabled or configured to accept requests only from the local system. While implementing this approach with traditional `rc` utility commands is certainly possible, it is not a very flexible approach, because only a single configuration can be captured on one system at any given time. Rather, Secure by Default uses the Solaris Service Management Facility (SMF) to control the affected network services. The key elements are the conversion of some existing services to SMF control, the addition of properties for existing SMF services to provide for local-only operation, the creation of an SMF profile to configure the system in the hardened state, and the `netserives(1M)` command to apply the SMF profile and set related SMF properties.

3 The Service Management Facility

The Service Management Facility (SMF) provides an infrastructure that amends the traditional UNIX start-up scripts, `init` run levels and configuration files. The SMF provides the following functionalities. It is described in great detail by Adams et al. in [1].

3.1 Service Management

Services can be enabled, disabled, or restarted using the `svcadm(1M)` command. Failed service(s) are restarted automatically in dependency order, whether the failure is a result of administrative error, software bugs, or because of an uncorrectable hardware error. Service objects can be viewed and managed with commands such as `svcadm(1M)`, `svcs(1)`, `svccfg(1M)`. Service failures are also relatively easy to debug as SMF provides the explanation of why a service is not running by using `svcs` command. Separate persistent log files make this process even easier. It is simple to restore, backup and undo changes to services by taking snapshots of service configuration files. Systems managed with SMF

boot and shut down much faster than traditional Unix systems, because services are started/stopped not sequentially, but concurrently, according to the dependencies between services. With the use of role-based access control (RBAC), administrators can securely delegate tasks to non-root users who can modify properties as well as status of services without having administrative privileges that go beyond the tasks at hand. Finally, SMF is compatible with traditional UNIX rc scripts.

3.2 Service Instances and Properties

SMF has a fundamental unit of administration known as a service instance. A service instance is a specific configuration of a service. For Example: A web server is a service and a single web server daemon that is configured to accept web service requests on port 80, can be called as an instance. Each SMF service can have multiple versions of it configured and multiple instances of the same version can run on a single Solaris operating system instance.

To name service instances, each service is associated with a Fault Management Resource Identifier (FMRI) that includes the service name and an instance name. For example, the FMRI for ssh is `svc:/network/ssh:default` where `network/ssh` denotes the service and `default` identifies the service instance. Legacy scripts are also represented with FMRI, however, they start with `lrc` instead of `svc`. For example, the legacy ppp daemon can be monitored using SMF via the FMRI `lrc:/etc/rc2.d/S47pppd`.

In SMF, service properties are defined in an XML format in files called SMF service manifests. They are stored in the directory `/var/sc/manifest`. The manifests are the authoritative source of configuration information of a service and their running instances and they can be modified with the SMF administrative tools, such as `svccfg(1M)` or `inetadm(1M)`.

3.3 Service Profiles

An SMF profile is a definition of which services should be enabled or disabled. It is represented in an XML file format. A few profiles are delivered by default with Solaris and are available in directory `/var/svc/profile/`. Administrators can customize copies of these profiles or define their own.

During the first boot after a new installation or an upgrade to the Solaris 10 release, some Solaris profiles are automatically applied. By default `/var/svc/profile/generic.xml` is applied. This file is usually symbolically linked to `generic_open.xml` or `generic_limited_net.xml`. If a profile called as `site.xml` is in directory `/var/svc/profile`, the contents of the profile are applied. The initial set of enabled services may be customized by an administrator at any given point of time.

4 Security Advantages Using the Solaris Service Management Facility

This section presents how the Service Management Facility is used as a fundamental building block for system security. SMF ties together a number of security

technologies to accomplish security goals, such as improving system and service availability, integrity assurance, resilience against attacks, administrative authorizations, and audit.

4.1 Improved Availability

SMF monitors if services that are under its control are properly executing. Failed services are automatically restarted in dependency order, whether they failed as a result of administrative error, software bugs, or uncorrectable hardware errors. In many cases restarting the service will solve the problem. In cases where it does not, the affected service, as well as its dependent services, are put into maintenance mode and an administrator can be notified. Services may consist of zero, one, or many processes. Firstly, services consisting of one process make it easy to understand the process to service mapping. If, e.g., the service process crashes, the service is down and needs to be restarted. Secondly, Sendmail is a good example of a service that consists of typically two processes. Failure of either process should cause sendmail to be restarted. Traditional monitoring implementations that relied on parent-child relationships to monitor process health break down, because in the case of sendmail both processes are re-parented to the init process. Thirdly, there is a category of processes that are transient processes. They appear briefly during the startup process or exist only to execute a few commands to affect configuration change.

SMF introduces a kernel interface called a contract. A process contract is used by userland processes to register an interest in a process and all its children. Whenever important state changes occur, the interested process is notified of those changes and can take corrective action. Such changes include process exit, coredumps, fatal signals, hardware errors, etc. Service restarters are then responsible for managing service instances in response to those monitored state changes. There is a comprehensive state model for managing services, which are always in one of the following states: uninitialized, disabled, offline, online, degraded, or maintenance. Service transitions happen because of administrative action or service errors. They are also influenced by service dependencies, a major step forward in complex system service administration, because through dependency information the root cause of problems is determined automatically. A system administrator is directly pointed to the component that must be repaired, which is not necessarily the one exhibiting the most visible failure. The command `svcs -x` is a powerful diagnostic tool, as it describes what is going on in plain language with simple output, usually including a URL where to get detailed information how to proceed.

In many cases, an administrator never needs to get involved and services are restarted by the service restarter that monitored their health via the service contract. These contracts are the generic mechanism to express the relationship between a process and the kernel-managed resources a process depends upon. There are systems with sophisticated hardware error handling capabilities (8). SMF can take advantage of such capabilities. Where formerly the only corrective action was to kill affected processes and risk cascading failure, or restart the

entire system, now the operating system can determine e.g., the broader effect of a faulted cell on a DIMM and manage the error flow between services gracefully.

4.2 Process Rights Management Integration

The Service Management Facility also serves as a launch pad for software to be transparently started under the Solaris privilege process rights management. Ideally, one would modify the source code of services to take advantage of the Solaris privilege mechanism, properly minimizing the privilege sets to what is required, including dropping privileges that are not needed, bracketing privileged operations, and relinquishing privileges that are no longer needed.

However, in many cases it is not possible to make such source code modifications, e.g., when using third party software that is distributed only in binary form. It is therefore important to have a mechanism that can approximate the least privilege behavior for services in this category. SMF provides just that, a way to launch services with a reduced set of privileges, thereby eliminating the need for binary changes for participation in the Solaris least privilege process rights management model. The use of SMF for operating services in privilege-aware mode is, however, functionally limited, as it is not possible to bracket individual operations or to relinquish privileges once they have been used and are no longer needed by the service.

Still, restricting a service's privileges out of over 70 to just the few that are needed, represents a major step forward, mitigating effects of future flaws and protecting data and service functionality from both faults and malicious behavior.

To configure services to run with reduced privileges, the manifest author or the administrator would set the appropriate properties as part of the service manifest using the `svccfg(1M)` command. The following example shows how to use the `svccfg` command to reduce privileges and the limit privileges for the `apache2` web service. It also shows a few other properties, such as the working directory and project and resource pools. The latter two are especially interesting from a security point of view, because they are the tie-in to the Solaris resource management. By setting those properties, it is possible to limit the amount of system resources available to applications, preventing system denial of service attacks, should individual services be attacked.

In the example in Table [1](#) the effective set of privileges for the web server are set to be the basic set (`file_link_any`, `proc_exec`, `proc_fork`, `proc_info`, `proc_session`), minus the privileges `proc_session`, `proc_info`, and `file_link_any` plus the `net_privaddr` privilege. The latter is needed for binding to a low-numbered port 1-1023 (usually port 80 for a web server.) The resulting effective set of privileges for this web server SMF service instance configuration is therefore just `proc_exec`, `proc_fork`, and `net_privaddr`. Were there access to the source code, one could tighten the webserver execution even further by

- bracketing the privileges `proc_exec`, `proc_fork`, and `net_privaddr` around the associated system and library calls `exec(2)`, `fork(2)`, `bind(3socket)`, and by

Table 1. Example how to reduce privileges for the apache2 web service through SMF.

```

svcadm$ svccfg -s apache2
svc:/network/http:apache2> setprop start/privileges = astring: basic, \
!proc_session, !proc_info, !file_link_any, net_privaddr
svc:/network/http:apache2> setprop start/limit_privileges = astring: \
basic, !proc_session, !proc_info, !file_link_any, net_privaddr
svc:/network/http:apache2> setprop start/working_directory = astring: \
default
svc:/network/http:apache2> setprop start/project = astring: default
svc:/network/http:apache2> setprop start/resource_pool = astring: default
svc:/network/http:apache2> end
svcadm$ svcadm refresh apache2

```

- relinquishing these privileges once the program flow makes them unnecessary to keep around. E.g., the `net_privaddr` privilege could be relinquished after the only call to `bind()`.

4.3 SMF Profiles

The information associated with a service or service instance that is stored in the configuration repository can be exported as XML-based files. Such XML files, known as service bundles, are portable and suitable for backup purposes. Service bundles are classified as one of the following types:

manifests Files that contain the complete set of properties associated with a specific set of services or service instances.

profiles Files that contain a set of service instances and values for the enabled property on each instance.

The Solaris Secure by Default stance is defined as an SMF profile. It reduces the overall system attack surface of the Solaris operating system by disabling as many network services as possible while still leaving a useful system. In this way, the number of exposed network services (in a default configuration) is dramatically reduced. The default configuration of the Solaris operating system is changed such that `ssh` is the only network-listening service. Other network services are either disabled or configured to accept requests only from the local system.

There is one high-profile example, where this mechanism helped save the day. In March 2007, when Sun distributed Solaris operating source code to external developers and users under an OpenSolaris license, an early scrutinizer of the code reported a previously unknown vulnerability in Solaris' telnet program. Because the code was open sourced, the report could and did include a pointer to the lines in the code that were responsible. The report was posted on a Sunday afternoon. By Monday, Sun engineers and testers in Australia, the UK, and the U.S. determined how the `telnet -f -l root` command could compromise a system, posted a Sun Alert, and provided a temporary fix. By Tuesday, Sun issued a patch. Sun's countermeasures were not complex. Between 1988 and today, Sun's

security features limit the damage an attack can inflict. A Secure By Default (SBD) installation closes the telnet port as part of the netservices limited SMF profile. And even if the port was open, a simple SMF command (`svcadm disable telnet`) ends the telnet service persistently across reboots.

The creation of new SMF profiles can be used to minimize or customize systems in deployment. Ideally one would combine this step with additional hardening by removing (or not even installing in the first place) software packages that won't ever be needed on a given system.

4.4 RBAC Integration

SMF uses the Solaris Role-Based Access Control (RBAC) facility to delegate access to SMF administrative tasks. By integrating with RBAC in this way, a consistent approach for privilege delegation is applied throughout the system, both if a command is run by a user or role who has been given specific authorizations or privileges or if a service is started by SMF directly. The RBAC authorizations `solaris.smf.modify` and `solaris.smf.manage` are the primary authorizations for granting additional privileges to users or roles.

There are also a number of property group-specific authorizations to provide additional granularity. They can be used to subdivide the authority of the `solaris.smf.modify` authorization for finer control. The authorization `solaris.smf.modify.method` permits changing values or creating, deleting, or modifying a property group of type `method`. Each service or service instance must define a set of methods that start, stop, and (optionally) refresh the service. The authorization `solaris.smf.modify.dependency` permits changing values or creating, deleting, or modifying a property group of type `dependency`. Service instances may have dependencies on services or files. Those dependencies govern when the service is started and automatically stopped. The authorization `solaris.smf.modify.application` permits changing values or creating, deleting, or modifying a property group of type `application`. This property group is reserved to store application-specific properties. Finally, the authorization `solaris.smf.modify.framework` permits changing values or creating, deleting, or modifying a property group of type `framework`.

These capabilities can be leveraged by using the RBAC rights profiles "Service Management" or "Service Operator" that are supplied with Solaris out of the box.

To run services directly through SMF with certain authorizations, the administrator would install the appropriate authorizations or properties as part of the

Table 2. Example how to reduce leverage RBAC controls for the apache2 web service through SMF

```
svcadm$ svccfg -s apache2
svc:/network/http:apache2> setprop httpd/value_authorization = \
  astring: solaris.smf.value.application.http/apache2
svc:/network/http:apache2> end
svcadm$ svcadm refresh apache2
```

service manifest using the `svccfg(1M)` command. The example above in table 2 shows how to use the `svccfg` command to install the `httpd/value_authorization` property for the `apache2` web service.

There is development guidance for SMF services to provide service related RBAC authorizations, as appropriate, by providing service-specific values for the `action_authorization`, `modify_authorization`, `value_authorization`, and `read_authorization` of property groups. All service instances bundled with Solaris include an appropriate set of service profiles. Service method scripts are delivered as read-only and customizable settings are accessible only via service-specific properties or existing configuration files. Sun delivered services also include an appropriate RBAC profile and authorizations for manipulating the service and its specific configuration.

5 Conclusions

The Solaris Service Management Facility (SMF) ties together a number of security technologies to accomplish security goals, such as improving system and service availability, integrity assurance, resilience against attacks, administrative authorizations, and audit.

The SMF improves service availability through automatic correction of failed services in dependency order. As a launch pad for third party software it transparently starts system services under the Solaris least privilege process rights management without the need to modify source code. The Solaris "Secure by Default" stance is implemented by defining a profile that configures new system installations to have only a single network-facing port open for ssh-based system administration. Strong audit and full administrator accountability are achieved through the tight integration of SMF into the Solaris administration role-based access control model.

References

1. Adams, J., Bustos, D., Hahn, S., Powell, D., Praza, L.: Solaris Service Management Facility: Modern System Startup and Administration. In: Proceedings of the 19th Large Installation System Administration Conference, LISA (2005)
2. Brunette, G.: Limiting Service Privileges in the Solaris 10 Operating System. Sun BluePrints OnLine (May 2005)
3. Faden, G.: Authorization Infrastructure in Solaris. Sun Developer Network (2001)
4. Ferraiolo, D.F., Kuhn, D.R.: Role-based access control. In: Proceedings of the 15th National Computer Security Conference, NCSC (1992)
5. Mewburn, L.: The design and implementation of the netbsd rc.d system. In: Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, pp. 69–79. USENIX Association, Berkeley (2001)
6. Pechanec, J., Schuba, C., Phalan, M.: New Security Features in OpenSolaris and Beyond. In: Proceedings of OpenSolaris Developer Conference, Prague, Czech Republic (2008)

7. Sangster, P., Bubb, V., Belgaied, K.: The Solaris Cryptographic Framework. In: BigAdmin System Administration Portal (2005), http://www.sun.com/bigadmin/features/articles/crypt_framework.pdf
8. Shapiro, M.W.: Self-healing in modern operating systems. *Queue* 2(9), 66–75 (2005)
9. Skinner, G., Czajkowski, G., Hearnden, D., Jordan, M., Wegiel, M.: A service management facility for java platform. In: SCC 2005: Proceedings of the 2005 IEEE International Conference on Services Computing, pp. 198–207. IEEE Computer Society, Washington (2005)

IntFinder: Automatically Detecting Integer Bugs in x86 Binary Program

Ping Chen¹, Hao Han¹, Yi Wang¹, Xiaobin Shen², Xinchun Yin²,
Bing Mao¹, and Li Xie¹

¹ State Key Laboratory for Novel Software Technology, Nanjing University,
Department of Computer Science and Technology, Nanjing University, Nanjing 210093
{chenping,wangyi}@sns.nju.edu.cn, {maobing,xieli}@nju.edu.cn

² College of Information Engineering, Yangzhou University,
Yangzhou Jiangsu 225009, China
xcyin@yzu.edu.cn

Abstract. Recently, Integer bugs have been increasing sharply and become the notorious source of bugs for various serious attacks. In this paper, we propose a tool, IntFinder, which can automatically detect Integer bugs in a x86 binary program. We implement IntFinder based on a combination of static and dynamic analysis. First, IntFinder decompiles a x86 binary code, and creates the suspect instruction set. Second, IntFinder dynamically inspects the instructions in the suspect set and confirms which instructions are actual Integer bugs with the error-prone input. Compared with other approaches, IntFinder provides more accurate and sufficient type information and reduces the instructions which will be inspected by static analysis. Experimental results are quite encouraging: IntFinder has detected the integer bugs in several practical programs as well as one new bug in `slocate-2.7`, and it achieves a low false positives and negatives.

1 Introduction

Integer bug is a notorious bug in programs written in languages such as C/C++, and it can not be obviously witnessed by traditional bug detector. Integer bugs can be classified into four categories: integer overflow, integer underflow, signedness error and assignment truncation. Recently, Integer bug has been increasing sharply, Common Vulnerability and Exploit (CVE) shows that integer overflow and signedness error have been increasing [7]. Besides, in OS vendor advisories, integer overflow has become the second most common bug type among all the bugs.

In recent years, several researches are focused on Integer bugs. Given program source code, there are three approaches to detect Integer bugs. (1) Safe Language, this method either translates the C program into type safe language (such as CCured [21], Cyclone [16]) or uses safe class (such as SafeInt [17], IntSafe [15]). (2) Extension to C compilers, this method inserts checking code for certain operations when compile the source code (such as BLIP [14], RICH [9]). (3) Static source code analysis, this method inspects the whole program to find the suspect instruction (such as LCLint [13], integer bug detection algorithm proposed by Sarkar et al. [11]). Although the works mentioned above may be effective for detecting Integer bugs, they all need source code,

which is not available for COTS programs. To prevent integer bugs at binary level, several techniques are proposed, such as UQBTng [25], BRICK [10], IntScope [24] and SmartFuzz [20]. UQBTng [25] is a tool capable of dynamically finding integer overflow in Win32 binaries. BRICK [10] is a binary tool for run-time detecting integer bugs. IntScope is a static binary analysis tool for finding integer overflow. SmartFuzz generates test cases that cause Integer bugs at the point in the program where such behavior could occur. The state-of-the-art techniques of detecting Integer bugs at binary level have several limitations. First, some tools, such as IntScope and UQBTng, can only detect integer overflow. Second, although these works render type reconstruction by using hints from the x86 instruction set, they do not explicitly discuss how to extract this information, and no approach proposes type extraction based on both data-flow and control-flow analysis. Third, the binary static method like IntScope has false positives, because it is difficult to statically reason about integer values with sufficient precision. IntScope leverages symbolic execution only for data that from outside input to the sinks defined by the tool, however, it lacks the information of constraints between inputs and global information [24]. By contrast, the dynamic binary approach such as BRICK [10] and UQBTng [25] may be time-consuming or have high false negatives, because it either checks all the suspect operations including the benign ones or ignores several critical operations. SmartFuzz can generate test cases to trigger Integer bugs, however, without dynamic detecting tools specific to Integer bugs, just using the memory error detecting tool such as Memcheck [22], SmartFuzz would not report certain test cases, which trigger Integer bugs but can not be detected by Memcheck. To sum up, existing binary level detection tools have at least one of the following limitations: (1) ineffective for integer bugs except Integer overflow (2) imprecise type reconstruction (3) high false positives of static analysis (4) time-consuming and high false negatives of dynamic analysis.

To overcome the limitations mentioned above, we implement our tool at binary level, and automatically detect Integer bugs by using static and dynamic analysis. Our paper makes three major contributions:

- We reconstruct the sufficient type information from binary code.
- We propose a systematic method of combining static and dynamic analysis to specifically detect integer bugs in executables.
- We implement a prototype called IntFinder and use it to analyze real-world binaries. Experimental results show that our approach has low false positives and negatives and is able to detect 0-day integer bugs.

The rest of this paper is organized as follows: The characteristic of Integer bugs are described in section 2. In section 3, we present the overview of IntFinder. The implementation of IntFinder is illustrated at section 4. Section 5 provides the evaluation of our tool. Section 6 examines its limitations. Finally, section 7 concludes our work.

2 Integer Bugs

We studied 350 Integer bugs on CVE [5], and noticed that the common root cause of Integer bugs is the mismatch of operand value and its type. We also noticed that

exploiting Integer bugs leverages several common mechanisms, which can also be used for us to detect the integer bugs. We summarize the most common tricks as following.

- *Memory Allocation Function*: Memory allocation functions directly deal with the memory space and certainly become the first target for attacker. As the size argument of memory allocation function is unsigned integer. There are two mechanisms to achieve attack. First, through integer overflow/underflow, the size parameter will get smaller or larger value than expected, and the allocated memory will be either insufficient or exhausted. Second, signedness error may lead to bypass the comparison operation, and transfer a negative value to the size argument of memory allocation function.
- *Array Index*: Array index can be regarded as an unsigned integer, and it is often leveraged to compute the offset from the base address to access the memory. When the array index is crafted by Integer bugs, it will facilitate the attacker to access any memory space.
- *Memory Copy Function*: Memory copy function often has the unsigned parameter, and the parameter determines the size of memory which is copied from source operand to destination operand. If the parameter is crafted by Integer bugs, it will lead to buffer overflow.
- *Signed Upper Bound Check*: Signed upper bound check (often comparison operation) can be bypassed by negative values. And later this value will be converted to be a larger values via a signedness error or integer underflow. If the large value is used as an argument of memory allocation or memory copy functions, an attacker may be able to exploit this bug to corrupt application memory.

3 Overview

In this section, we will describe the architecture and working process of IntFinder. There are three main components in IntFinder: (1) Extended type analysis on decompiler (2) Taint analysis tool (3) Dynamic detection tool. Figure 1 shows the architecture.

The work flow of our approach is as below: first, IntFinder leverages the decompiler to translate x86 binary program into its SSA-like intermediate language. Second, we

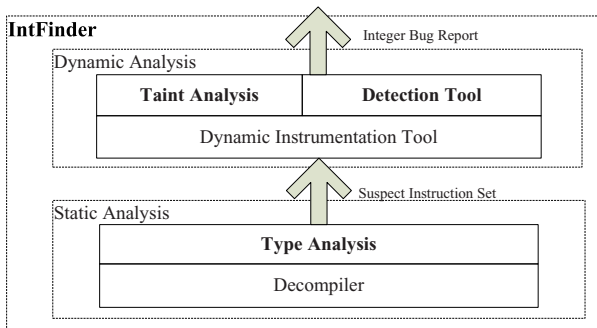


Fig. 1. Architecture of IntFinder

Table 1. Example of control flow based type information

```

[1]   char * buf;
[2]   int size;
[3]   if (...) {
[4]     if (size > 100)
[5]       {
[6]         ...
[7]       }
[8]     else
[9]       ...
[10]  }
[11]  else {
[12]    ...
[13]    buf = (char *) malloc (size);
[14]  }

```

extend the type analysis of decompiler in order to get more type information from binary code and create the suspect integer bug set. Third, with the suspect integer bug set, we implement the dynamic detection tool combined with taint analysis to further diagnose the instructions in the suspect sets, and determine which are Integer bugs.

- *Type Analysis*: Our type analysis is based on decompiler. It provides the relatively precise type information and gives the suspect instruction set for dynamic testing or debugging. Compensating to existing type information extracted by decompiler, IntFinder further rebuilds the type information based on exploited function and statement just discussed in Section 2. In addition, IntFinder also reconstructs the type information by using both data flow analysis and control flow analysis. Control flow analysis will be benefit because type information of operand may be collected from different basic blocks. Table 1 shows an example of control flow related type information, which contains apparently type conflicting about signed integer variable named *size*, and has potential signedness error. Specifically, at line 4 *size* is illustrated as signed integer, while at line 13 *size* will be illustrated as unsigned integer. Note that statements at line 4 and line 13 are at different basic blocks. Thus, conflicting type information may be missed without control flow analysis.
- *Taint Analysis*: To further reduce the number of suspect instruction and false positives of our tool, we use dynamic taint analysis to select those instructions with tainted data.
- *Dynamic Detection Tool*: Our detection tool is implemented on dynamic instrumentation tool. During program is executing, the tool selects the suspect instruction which is tainted and uses our checking mechanisms to verify whether it is a real Integer bug.

4 Implementation Details

In this section, we give the detailed design of IntFinder, which is implemented on decompiler Boomerang-0.3 [12] and dynamic instrumentation tool PIN-2.2 [19].

4.1 Type Analysis

Type Information Extraction. As discussed in Section 3, our type analysis is implemented on decompiler. To get sufficient type information, we extend type analysis on Boomerang from the following three aspects:

- Existing integer type reconstruction of Boomerang only considers some specific arithmetic operation, such as IMUL and SAL, to determine the signedness type of operand, we additionally extract the type information at several sensitive points just as discussed in Section 2.
- Boomerang provides the sparse type information, that is to say, it does not store the type information for operand. To be convenient for type analysis, we additionally store type information for memory operand in each statement. In addition, Boomerang does not consider the backward type information propagation (in the reverse order of normal decompilation). However, for Integer bugs, we need to trace back to determine the previous undefined type. For example, in signedness error, we need to find whether current type conflicts with the previous type information.
- Existing type analysis of Boomerang only extracts the type information based on data flow analysis, and it can only extract type information within single basic block. We traverse the control flow graph, and propagate the type of operand to other control branch and eventually provide sufficient type information.

We rebuild the type information in the following steps: First, at loading time, we modify the signature of some library functions. For example, we modify the type of the memcpy’s third argument from “*size_t*” to “*unsigned int*”. Second, decompiler executes its own type analysis. Third, at certain functions and statements discussed in Section 2, we extract additional type information and propagate the type information backwards within a basic block. Fourth, we traverse the control flow graph, and propagate the type information to other basic blocks. Note that we only propagate the type with sufficient information (contains both signedness and width type). When we find the type conflicting, we set the type of the operand as “*BOT*”, which is not propagated. We stop traversing the control flow when there is no updated type information.

Suspect Instruction Set. After type analysis, we traverse the intermediate statements generated by decompiler, and create the suspect instruction set. The format of suspect instruction is shown in Table 2. *Address* field is the address of suspect instruction. *bug-type* field is the type name of the bug. *opcode* field is specific to integer overflow/underflow, including arithmetic operations. *signedness type* field is the signedness type of the destination operand in the instruction. *type of left operand* field is the type of left operand, including memory, register and constant value. *type of right operand* field is the type of right operand. *size* field is the width type of the destination operand.

Table 2. Format of suspect instruction set

Address	bugtype	opcode	signedness type	type of left operand	type of right operand	size
---------	---------	--------	-----------------	----------------------	-----------------------	------

4.2 Taint Analysis

We find that Integer bugs often exist in the applications which get input resources just like network package, configuration file, database file, user command and so on. Some typical source functions manipulate the input data, like `read`, `fread`, `recv` and so on. We select these functions as the source of taint analysis, and tag the memory which holds the data from these functions, and then, we propagate the tag according to different kinds of instructions in the granularity of byte-wise. We only check those suspect instructions whose operands are tainted. Taint analysis can help us to distinguish the malicious instruction from the benign instruction which is often introduced by programmer or compile optimization, and then reduce the false positives.

4.3 Dynamic Detection Tool

Our detection tool is implemented on dynamic instrumentation tool PIN [19], it leverages the suspect instruction set produced by type analysis and apply our checking scheme to these instructions. Our checking scheme can be divided into three categories:

- *Integer Overflow/Underflow*: We check integer overflow/underflow at arithmetic operations. Followed the integer overflow detecting rule introduced by William Stallings [23], we determine integer overflow/underflow by using EFLAGS register. However, there are several exceptions that we need to re-calculate the result of the arithmetic operation. Take 8/16 bits addition for example, GCC compiler promotes the 8/16 bits operand to 32 bits register, and then do the addition operation. It will fill dirty value to the high part of the register, and finally set the *CF* or *SF* flag incorrectly. Instead, we re-calculate the addition for these two cases.
- *Signedness Error*: We check the value of the operand which has the conflicting type information, and determine whether it has a negative value.
- *Assignment Truncation*: When truncation occurs, the value of the source operand will be larger than the maximum value which destination operand can hold, then we check whether the high-order bits of source operand are not zero.

5 Evaluation

We evaluated IntFinder with several utility applications. The evaluation is performed on an Intel Pentium Dual E2180 2.00GHz machine with 2GB memory and Linux 2.6.15 kernel. Tested programs are compiled by gcc 3.4.0 and linked with glibc 2.3.2.

5.1 Suspect Instruction Set

As discussed in Section 4, IntFinder statically selected the suspect instructions, it is the first step to reduce the instructions to be checked. Table 3 shows the number of suspect instructions produced by type analysis. We can see that, in average, IntFinder statically reports about six suspect instructions per 100k. In order to evaluate the precise of type reconstruction, we compare the type information listed in suspect instruction set to original source code manually. As shown in Table 3, we find that the precise of type

Table 3. Suspect Instruction Set of IntFinder

Program	Size	Overflow/Underflow	Signedness Error	Truncation	Total	Precise
PHP-5.2.5	10.3M	330/339	84/84	234/234	648/657	98.6%
slocate-2.7	46.8K	6/6	1/1	1/1	8/8	100%
zgv-2.8	284.7K	23/24	19/19	16/16	58/59	98.3%
python-2.5.2	3.4M	95/96	21/21	61/67	177/184	96.2%
ngircd-0.8.1	329.1K	15/17	13/13	18/19	46/49	93.9%
openssh-2.2.1	150.8K	14/15	1/1	9/9	24/25	96%

Precise rate of type reconstruction is in the form of x/y , “x” represents the instructions with precise type, and “y” represents the total suspect instructions.

reconstruction is nearly to 100%. Note that some false positives for type reconstruction exists in IntFinder, because it is hard to distinguish the following two cases: (1) pointer and integer variable; (2) pointer and array. The former case may cause some pointer arithmetic operation being regarded as potential integer overflow/underflow. The later case may cause the imprecise type analysis for the offset of pointer, because the offset of pointer should be regarded as signed integer, which is different from the unsigned type of the array index. However, we consider that they are non-trivial problems.

5.2 Analysis of False Positives and False Negatives

We choose several real applications to verify whether integer bug can be efficiently detected. The experiment results in Table 4 and Table 5 show that IntFinder has low false negatives and false positives.

We also test the false positives of IntFinder. Note that we uses the same error-prone input as used in false negatives testing. Table 5 shows that the false positives of IntFinder is relatively low. There is no false positives in our experiments.

Table 4. Applications with Integer Bugs Tested on IntFinder

CVE#	Program	Vulnerability	Types	IntFinder
2008-1384	PHP 5.2.5	php_sprintf_appendstring() bug [6]	Integer Overflow	✓
2003-0326	slocate-2.7	parse_decode_path() bug [1]	Integer Overflow	✓
2004-1095	zgv-5.8	multiple integer overflow [2]	Integer Overflow	✓
2008-1721	python-2.5.2	zlib extension module bug [8]	Signedness Error	✓
2005-0199	ngIRCd-0.8.1	Lists_MakeMask() bug [3]	Integer Underflow	✓
2001-0144	openssh-2.2.1	detect_attack() bug [4]	Assignment Truncation	✓

Table 5. False Positives of IntFinder

CVE#	Program	Overflow/Underflow	Signedness Error	Truncation	Real Bugs
2008-1384	PHP-5.2.5	1	0	0	1
2003-0326	slocate-2.7	1	0	0	1
2004-1095	zgv-5.8	11	0	0	11
2008-1721	python-2.5.2	0	1	0	1
2005-0199	ngIRCd-0.8.1	1	0	0	1
2001-0144	openssh-2.2.1	0	0	1	1

5.3 Performance Overhead

We evaluate the performance overhead of IntFinder. Table 6 shows the slow down factors of IntFinder. Each program runs natively, under PIN without instrumentation and under IntFinder without taint analysis once respectively. We find that the average performance overhead of PIN without instrumentation is 3.7 X, while the average performance overhead of IntFinder without taint analysis is 4.4 X. Note that we also test the performance overhead of taint analysis, which is not list in Table 6. The average slow down factor of IntFinder with taint analysis is nearly 50X. We provide the interface of taint analysis for the user to choose open/close it.

Table 6. Performance Slow Down

Program	Benchmark	Native Run	Under PIN	Under IntFinder	Slow Down of PIN	Slow Down of IntFinder
PHP-5.2.5	CVE-2008-1384	0.022s	1.101s	1.432s	50.0 X	65.1 X
slocate-2.7	CVE-2003-0326	0.008s	0.296s	0.390s	37.0 X	48.7 X
zgv-5.8	CVE-2004-1095	0.101s	0.348s	0.447s	3.4 X	4.4 X
python-2.5.2	CVE-2008-1721	0.585s	2.033s	2.592s	3.5 X	4.4 X
ngIRCd-0.8.1	CVE-2005-0199	1.156s	3.945s	4.377s	3.4 X	3.8 X
Average		0.420s	1.545s	1.848s	3.7 X	4.4 X

5.4 New Bugs

IntFinder uncovered one new signedness error in slocate-2.7. This bug exists in `decode_db` function in `main.c`. In `slocate`, a signed integer “`tot_size`” is passed as the size argument to `realloc` at `main.c : 1224`, but this value may be negative after shift operation at `main.c: 1222`, it depends on the size of database file which reads from `main.c: 1232`. When the size of database reaches “G” level, it will trigger the integer bug. However, we could find no evidence in mailing lists or CVS logs that the developers were specifically trying to fix this bug.

6 Discussion

6.1 Decompile Limitation

- *Missing certain functions*: In experiment, decompiler Boomerang may fail to decompile some functions, we find that these functions either have indirect jump which may be hard for decompiler to construct accurate path, or have some relationship with other functions which can not be decompiled. Fortunately, from the experiment results, we find that the accurate decompile rate is above 95%.
- *Imprecise of decompiler*: Boomerang suffers from the common limitations of decompiler: (1) Imprecise of decoding the indirect jump. (2) Imprecise differentiate the pointer from integer variable. (3) Imprecise differentiate the pointer from array.

6.2 Dynamic Detection Limitation

- *Semantic Error* IntFinder may lose Integer bugs caused by semantic error. For example, NetBSD has an Integer bug and suffers from forcing a reference counter to wrap around to 0, which may cause the referenced object to be freed even though it was still in use.

- *Logic Operation* In our current implementation, we ignore the Integer bugs associated with certain logic operation, except the SHL and SAL. We have the following consideration: (1) It is hard to distinguish the benign logic operation from the malicious one. (2) Logic operation occurs frequently in the program. If we check the logic operation, it will bring more false positives and raise performance overhead.

6.3 Dynamic Testing Input Limitation

In our current implementation, we use existing error-prone inputs published by vulnerability database just like CVE. However, this method can only detect the known integer bugs. To further leverage our tool to find more unknown vulnerability, we need to construct the input associated with the suspect instructions. To achieve the goal, there are certain feasible methods: (1) Using symbolic execution to generate inputs to trigger the suspect instructions. (2) Using some verification tools, such as dynamic testing tool [18], to construct the relationship between input and suspect points.

7 Conclusion

In this paper, we present the design, implementation, and evaluation of IntFinder, a tool for automatically detecting integer bugs. Given a binary code, IntFinder decompiles a binary, and creates the suspect instruction set. Then IntFinder dynamically inspects the instructions in the suspect set and confirms which instruction is real Integer bug with the error-prone input. Compared with other approaches, IntFinder provides more accurate and sufficient type information and reduces the instructions which will be dynamically inspected. The evaluation of IntFinder shows that it has low false positives and negatives.

Acknowledgements

This work was supported in part by grants from the Chinese National Natural Science Foundation (60773171, 90818022, and 60721002), the Chinese National 863 High-Tech Program (2007AA01Z448), the Chinese 973 Major State Basic Program (2009CB320705), and the Natural Science Foundation of Jiangsu Province (BK2007136).

References

1. Integer overflow in parse_decode_path of slocate. CVE (2003), <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-0326>
2. Integer overflow in zgv-5.8. CVE (2004), <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-1095>
3. Integer underflow in ngircd before 0.8.2. CVE (2005), <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-0199>
4. Ssh crc-32 compensation attack detector vulnerability. CVE (2005), <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144>
5. Cve version: 20061101. CVE (2006), <http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=integer>

6. Integer overflow in pdftops. CVE (2007),
<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1384>
7. Vulnerability type distributions in cev. CVE (2007),
<http://cve.mitre.org/docs/vuln-trends/vuln-trends.pdf>
8. Signedness error in python-2.5.2. CVE (2008),
<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1721>
9. Brumley, D., cker Chiueh, T., Johnson, R., Lin, H., Song, D.: Rich: Automatically protecting against integer-based vulnerabilities. In: Proceedings of the 14th Annual Network and Distributed System Security, NDSS (2007)
10. Chen, P., Wang, Y., Xin, Z., Mao, B., Xie, L.: Brick: A binary tool for run-time detecting and locating integer-based vulnerability. In: International Conference on Availability, Reliability and Security, pp. 208–215 (2009)
11. Dipanwita, S., Muthu, J., Jay, T., Ramanathan, V.: Flow-insensitive static analysis for detecting integer anomalies in programs. In: Proceedings of the 25th conference on IASTED International Multi-Conference (SE 2007), pp. 334–340. ACTA Press, Anaheim (2007)
12. Emmerik, M.J.V.: Static Single Assignment for Decompilation. Ph.D. thesis (2007)
13. Evans, D., Guttag, J., Horning, J., Tan, Y.M.: Lclint: a tool for using specification to check code. In: Proceedings of the ACM SIGSOFT 1994 Symposium on the Foundations of Software Engineering, pp. 87–96 (1994)
14. Horowitz, O.: Big loop integer protection. Phrack Inc (2002),
<http://www.phrack.org/issues.html?issue=60&id=9#article>
15. Howard, M.: Safe integer arithmetic in c (2006),
<http://blogs.msdn.com/michaelhoward/archive/2006/02/02/523392.aspx>
16. Jim, T., Morrisett, G., Grossman, D., Hicks, M., Cheney, J., Wang, Y.: Cyclone: A safe dialect of c. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference (2002)
17. LeBlanc, D.: Integer handling with the c++ safeint class (2004),
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure01142004.asp>
18. Lin, Z., Zhang, X., Xu, D.: Convicting exploitable software vulnerabilities: An efficient input provenance based approach. In: Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN-DCCS 2008 (2008)
19. Luk, C.K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., Wallace, S., Reddi, V.J., Hazelwood, K.: Pin: Building customized program analysis tools with dynamic instrumentation. In: Programming Language Design and Implementation, pp. 190–200 (2005)
20. Molnar, D., Li, X.C., Wagner, D.: Dynamic test generation to find integer bugs in x86 binary linux programs. In: Proceedings of the USENIX 2009 Annual Technical Conference (2009)
21. Necula, G.C., McPeak, S., Weimer, W.: Ccured: Type-safe retrofitting of legacy code. In: Proceedings of the Principles of Programming Languages, pp. 128–139 (2002)
22. Seward, J., Nethercote, N.: Using valgrind to detect undefined value errors with bit-precision. In: Proceedings of the annual conference on USENIX Annual Technical Conference (2005)
23. Stalling, W.: Computer organization and architecture designing for performance. Prentice Hall, Inc., Englewood Cliffs (1996)
24. Wang, T., Wei, T., Lin, Z., Zou, W.: Intscope: Automatically detecting integer overflow vulnerability in x86 binary using symbolic execution. In: Proceedings of the 16th Annual Network and Distributed System Security Symposium, NDSS 2009 (2009)
25. Wojtczuk, R.: Uqbtng: a tool capable of automatically finding integer overflows in win32 binaries. In: 22nd Chaos Communication Congress (2005)

A Comparative Study of Privacy Mechanisms and a Novel Privacy Mechanism [Short Paper]

Gunmeet Singh and Sarbjeet Singh

University Institute of Engineering and Technology, Panjab University
Sector-14 Chandigarh, India
gunmeetsingh@gmail.com, sarbjeet@pu.ac.in

Abstract. Privacy of PII(Personally Identifiable Information) on the Internet is a major concern of a netizen. On the Internet different service providers are supposed to publish their own privacy policies but understanding of these policies is a major problem. Standards like Platform for Privacy Preferences(P3P), provide a computer readable format and a protocol for allowing web browsers to retrieve and process privacy policies. In this paper we studied the various privacy mechanisms in place and compared them on the basis of their architecture and third party intervention. We also proposed an alternative privacy mechanism that introduces the concept of a third party whose role is to verify the privacy policy and keep a proactive check on the use of specified PII's. In case of a violation the third party, informs the users of the breach. The implementation of the proactive check on the PII has been done through software agents. The requirement of granting legal status to transactions of the PII by the use of Digital Signatures and PKI has also been proposed,thereby legally binding the web entity to use the PII as per the agreed terms.

Keywords: Privacy, Trusted Third Party, Security,P3P, EPAL, Digital Signatures, Personally Identifiable Information (PII), Security.

1 Introduction

The growth of web services which require the use of PII's has increased manifold. Hence the use and distribution of the PII's between business entities have increased. The misuse of the PII which has resulted in crimes like spim , spam and junk mails. As PII itself is an identity of a netizen on the Internet. Therefore the use of PII itself should be checked and verified by the user at the service providers end. The interchanging of the PII's between business entities should also be notified to the user. . In case the entities commit a misuse of the PII they should be legally held for such a breach of confidence. Paradoxically the netizen will express very strong concerns about privacy of their PII, but be less than vigilant about safeguarding it [1]. Thereby requiring the inclusion of third party to safeguard the PII.

Web Services are associated with a Privacy Policy which states the objectives and aims of using the PII's [2][3]. Trade practices require that the privacy policies should be stated by the web services and laws like GLBA(Gramm Beach Bliley Act) state that the language of the policy stated should be in very simple and in clear terms[3].But the users of these services do not use the policies as they find these policies complex and difficult to understand. This does not allow user to make an informed choice about sharing their PII's.W3C provided a mechanism which allows the web services to state their policies in XML encoded form called P3P(Platform for Privacy Preferences)[5].This XML encoded form makes it easier to understand and provide a standardized way of stating privacy policy. And as compared to the privacy policies stated in Human Readable languages ,it is much easier to understand. Different other mechanisms like E-P3P, EPAL provide and enforce privacy policies inside Web Service entity. These restrict the access to the PII's to different groups inside an entity. But this whole approach to privacy is an inactive one ,once the PII is given to the Web Service there is no check to how the data is being used. We propose a privacy mechanism that solves this problem by introducing the concept of trusted third party which monitors the use of the PII by the web service. And in case there is an unauthorized use of the PII which violates agreed terms between the user and the web service. Thereby keeping a proactive check on the use of the PII.

2 Different Privacy Mechanisms

Most of the transactions for any service e.g. Setting up an email account on a email service, on the Internet is never complete without the exchange of PII. Hence the need to ensure the privacy of the PII is very important and must be addressed technically. Hence various mechanisms were developed and adopted the most popular being the P3P and EPAL. P3P was the first to be introduced and helped the privacy policy of the company to be stated in the machine readable form. The next step is to allow user to specify its privacy preference with the P3P document of the service. This is implemented with the use of user agents which allow the comparison of the policy and the preference e.g. AT& T Privacy Bird[6].The natural extension to this is to enforce the privacy policy of the company through out the the organization. EPAL is the mechanism that implemented the privacy policy through out the organization. This allows transparency and the synchronization of the privacy policies and the internal PII usage practices.

2.1 P3P (Platform for Privacy Preferences)

P3P is a standard defined by W3C,that allows a Web Service to state the privacy policy in a standardized machine readable form. The privacy policy states all the objectives of a web service regarding the information collected in a semi-structured XML form. The P3P specification[5] has a standard vocabulary to describe data practices which states the entities that will access the data and the purpose. Base data Schema is used for collecting information. An overview of the P3P vocabulary as stated in the P3P Specification[5] is described in the Table 1.

Table 1. Overview of the P3P vocabulary

P3P Policy Element	Detail
Entity	The web service or website which collects the information. This element stores the contact information of the entity which will collect the users PII
Access	This element specifies the which sub-entities can access the PII
Dispute	Describes how to resolve related disputes with the web service
Data	The kind of data collected by the web service
Purpose	This element states the purpose for which the PII collected will be used
Recipient	States the entities with which the data will be shared
Retention	Describes the retention policies of the information collected
Consequences	Human Readable element and explains the web services data practices

The privacy policy in the P3P Specification uses the above vocabulary to state their privacy policy. The P3P specification also has a protocol, built on the HTTP protocol to transmit and receive the privacy policies

2.1.1 Overview of P3P

The Privacy Preferences is the XML format defined under P3P which provides for the user to state the privacy preferences and also provides the algorithms for matching of the user privacy preferences with the web services privacy policy which are stated in the W3C APPEL. The overview of P3P and the various steps involved are shown in the Figure 1.

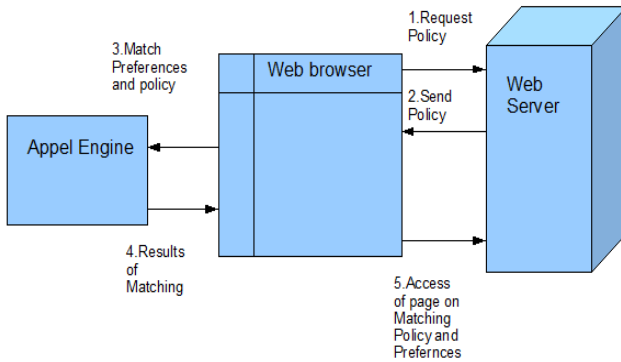







Fig. 1. Overview of P3P

2.1.2 Overview of User Agents

User Agents are tools that fetch the P3P policy of the site .The user agents like AT&T privacy bird,Microsoft IE 6 and Privacy Companion allow users to specify their privacy preferences[6].They can either be built into the web browser or can be end user applications. User agents also compare the privacy policies to privacy preferences and gives advice to users to whether to exchange PII with the specified web service. If the privacy policy is found in conflict with the user privacy preference the user agent takes the appropriate action e.g. blocking the page,Displaying a warning on the status bar. Hence allowing the user to make a choice to whether share the information e.g. AT&T Privacy Bird a very popular user agent uses the following Symbols are showed in the header of the Web Browser [6],[7] to show different status. Different symbols are described in Table-2.

Table 2. Symbols and Descriptions of AT&T Privacy Bird[6]

Header Symbols	Details
	This symbol indicates that the privacy policy and the preferences matches.
	Indicates that privacy policy and the preferences match but the site contains some frames, pictures etc.
	Indicates that Web Service\Site does not have P3P policy.
	Indicates that the Privacy policy does not Match
	Indicates that the Tool has been turned of

Thereby this gives adequate warning and information about the site to the user. Hence user can make a informed choice about the site.

P3P also provides mechanism for specifying cookie related data practices. These P3P policies are referred to as “compact policies”.These are included in HTTP-Response headers and provide a quick way for a user agent to compare the policy with the preferences without referring to another document.

2.1.3 Drawbacks of P3P

P3P though provides a machine readable format for the specification of privacy policy, but the mechanism to ensure the privacy of the user is not present. The P3P privacy policy is a formal document that states the usage of the PII, but it is not enforceable throughout the web service.P3P documents are difficult to write as compared to EPAL Policy documents[8].Due to its complicated syntax it has been not adopted widely[9].

2.2 EPAL(Enterprise Privacy Authorization Language)

EPAL[10] [11] is a XML based privacy specification language that is used by organizations to specify their internal privacy policies. EPAL is basically used by web services or entities to define how the data is accessed inside them, it also synchronizes the policies of the entities and their business partners so that compliance is assured between the respective business policies. Developed by IBM, it was defined as a formal language to specify internal privacy policies which unlike P3P are enforceable and automated across entities systems. The privacy policy in EPAL is made up of elements which are analogous to P3P's policies elements, these defines access to the data. These are shown in table 3.

Table 3. Overview of EPAL elements

EPAL Policy Element	Detail
Ruling	Specifies one rule which can be “allow” or “deny”
User Category	This element specifies which User Group can access the specified data
Action	Models how the collected data is used.
Data-category	The different kind of data collected by the web service e.g. medical record or Contact information
Purpose	This element states the purpose for which the PII will be used. Can serve as a ruling whether to allow or deny the use of PII
Obligation	This element states certain actions to be completed
Condition	This element defines external conditions

The EPAL policy stated is enforceable throughout the organization. This is done by the Enforcement Engine[10] which parses the policy stated and controls the access of the user groups to the data store. The overview of the EPAL is given below in the Figure 2.

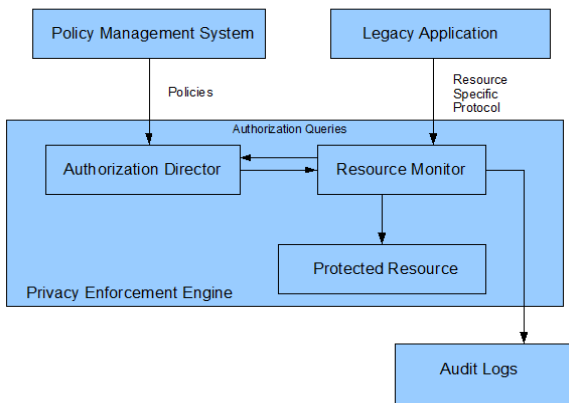


Fig. 2. Overview of the EPAL architecture

The EPAL policy is made up of rules which define the access conditions e.g. consider the EPAL policy given below

```
<epal-policy>
  <rule id="Email_Privacy_Rule_1" ruling="allow">
    <user-category refid="Subsidiaries"/>
    <data-category refid="Email"/>
      <obligation refid="Check opt-out list"/>
  </rule>
</epal-policy>
```

Consider the rule Email_Privacy_Rule_1. The ruling is “allow” which means allow access to the Email address. The user group allowed access is the Subsidiaries. The Obligation that the user group needs to perform is the checking of the opt-out list.

Another important concept of EPA called “Sticky Policy Paradigm” [10] which states that the terms and conditions which were promised or agreed upon will be applicable even if it is transferred from one entity to another.

2.3 Limitations of Current Privacy Mechanisms

The Privacy Mechanisms discussed above have the following disadvantages

1. No third party evaluation:- In both of the privacy mechanism there is no third party involved. The concept of third party ensures the impartiality of the system.
2. Non Proactive Approach towards protection of PII:-The PII is the identity of a netizen. And the user/netizen requires that the web service which has its PII to disclose the way it has been used. This can help in solving lots of problem related to proliferation of data from unscrupulous web services e.g. spam, spim, fraud.
3. Legal Status for PII transactions and its use:- PII is the identity of a netizen on the INTERNET .The privacy policies purpose is to ensure user that the PII will be safe. But to give legal recognition to this transaction of is the need of the hour.

Limitations are summarized in Table 4

Table 4. Limitations of the Current Privacy Mechanisms

Limitations	P3P	EPAL
Enforceability of privacy policy throughout the web service	No	Yes
Easily understandable syntax	No	Yes
Third Party Audit and Evaluation of PII use	No	No
Ability of user to monitor use of PII	No	No
Legal binding on exchange of PII	No	No
Legal Binding on the use of PII as per the stated privacy policy	No	No
Ability to link policy with data(Sticky Policy paradigm)	No	Yes

3 Proposed Privacy Framework

The proposed framework is a scheme for privacy-enabled management of netizen's data. Its core is an authorization scheme that defines how collected data may be used. Also a platform which facilitates third party evaluation is added to the proposed framework. The Proposed Privacy framework has four parties involved

1. User-The user represents a netizen who accesses and shares his/hers PII with the web service.
2. Web Service-The web service is an organization /entity which provides service and requires the PII from the user. Web service is also required to share some access details/logs with the trusted third party.
3. Trusted Third Party-The trusted third party is an entity which monitors the use of PII given to the web service for the user. This party legally binds the web services to follow the conditions agreed on by the user and the web service.
4. Controller For Privacy Insurers- This party will control the Trusted third party. It is a regulatory body and will control and regulate the trusted third party. Each country will have one. Since its role is similar to the CCA (Controller of Certifying Authorities) ,it can perform the role of CPI.

3.1 Prerequisites and Application Model

The Web Service run applications that collect and use PII's. Each application performs some tasks. For example a "view record" displays the record of a certain user.

Therefore there is a privacy policy that controls the access of the PII throughout the organization. Privacy policy may be informal rules that are applicable throughout the web service. These privacy policy is implemented in the form of EPAL policies in the system.

3.2 Policies Definition and Conversion

The Privacy policies state how the PII will be used. The privacy policy is defined using the EPAL Rules. These rules define the usage of a certain field in the form of information. The EPAL policy thus stated is thus converted into the P3P format [12] and hosted by the web service.

3.3 Collection and Transfer of PII

This is the first step when a user interacts with the web service for the first time and agrees to use the service and transfer the PII. The following steps take place

1. The user accepts the web services terms and conditions also stating the Opt-in and Opt-out choices and sends its privacy preferences in the XML form as stated in the privacy preferences in P3P.
2. The Web Service replies back to the user,the XML form is converted into a SQL query and it is compared with the web services privacy policy. The reason why the user sends the web service also returns its privacy policy.

- The user transfers the web services privacy policy to the trusted third party to analyze the privacy policy further. The trusted third party can store the history of the defaulter web services hence advice the user about the web services previous record
- 4. The trusted third party returns its analysis to the user.
- 5. The user and Web service transfers the PII and sign a contract which states the users preferences and the web services privacy policy with their private keys.
- 6. The document is further attested by the third party and a clause is added that the third party will be allowed full access to access details .The web service passes a “key” to the third party which will identify the specified PII in the web service.
- 7. The user is informed that the transaction is complete.

The overview of the collection and transfer of the PII is given in the Figure 3 below

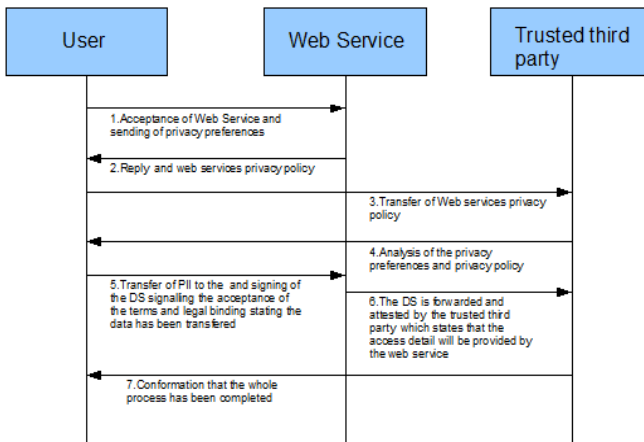


Fig. 3. Overview of the collection and transfer of the PII

3.4 Granting or Denying Access

The form and associated Opt In and Opt Out choices is used to decide whether the access will be granted or not. Any access to the PII is checked and verified as per the agreed terms and a separate audit log is maintained.

3.5 Privacy Insuring Mechanism

The privacy of the user is ensured by an evaluation of the audit logs that contains the access details to the PII. The access details includes the following fields of information

- 1. Access Purpose-States the purpose of PII access.
- 2. Access Time
- 3. Access Fields-The fields of information that were accessed
- 4. Request By-By whom the access request is generated. This field will store the information uniquely identifying user or the program that accessed the PII.

5. Sub-Entity name-Every employee of the web service is part of certain sub entity which can be the different departments of the company for e.g. Marketing,Finance.

The audit of the the access details is carried by a software agent which is deputed by the third party that was involved in the transfer of PII phase. The access detail of the specific PII is searched by a unique key that was transferred to the third party in the previous steps. The following steps are involved in whole process.

1. The user feels that the PII has been leaked e.g. unsolicited communication ,the user informs the trusted third party.
2. The Trusted third party asks the Web services for access details/logs. This step can occur at regular intervals of time. Inability of the web service to provide access to Trusted Third Party leads to the breach of contract. Hence making the Web Service accountable and need to be compliant .The access of audit Logs can be implemented by the use of Software Agents which is discussed later.
3. The Access detail is provided by the Web service to the Trusted Third Party.
4. The Analysis of the Access detail is done and the terms and conditions agreed by the user and the Web Service and check if any breach occurred. If any breach of terms agreed upon the user and the web service,user is informed and the user is entitled to take Legal remedies.

3.6 Access Classes

The access details reveal the day to day activities of the web service to the trusted third party. Therefore there is a abstraction layer required for hiding which sub entities access the PII's, at the same time not hiding the logs from the trusted third party.

This is implemented by Access Classes which classifies the different sub entities of the Web services into access classes. Classification is done as per the data proliferation risk each sub entity presents for e.g.

1. Least Data Proliferation Risk Class:-This class consists of sub business entities like the Maintenance Department,which requires very frequent access to the data and the risk for the proliferation of an individual PII is very less.
2. Medium Data Proliferation Risk Class:-This class consists of those entities that require less frequent access to the data or require data for purposes that include marketing.
3. High Data Proliferation Risk Class:-This class consists of the entities from where the proliferation of the PII is a major issue. This class should only contain entities that the organization wants to restrict the access of the PII's for e.g. the subsidiaries and other partner companies lie in this class.

The exact sub entity in the access details is mapped to the access class thus protecting the integrity of web services.

3.7 Framework Overview

The whole framework is Consists of the following Sub Modules

1. Privacy Policy Sub-Module:-This sub module states the privacy policy of the organization and helps in stating the Privacy policy which can be converted from

the Internal Privacy Policy as stated by Karjoth et AL (Conversion of EPAL[12] to P3P).This Submodule will produce a P3P document that will state all the data usage terms which the user can compare with his/her privacy preferences. Thus this Sub Module will represent the privacy system to the user when he first accesses the web services and wants to transfer his/her PII.

2. Digital Signature Storage Sub-Module:-This Module stores the agreements signed by the user and the trusted Third party with the web service .This component will act as a repository and since the UNICITRAL's Model IT law the Digital Signatures has the same status as a signed contract this protects the web service from any fraud and gives legal recognition to transfer to the PII.
3. Internal Privacy Enforcement Engine:This engine is similar to EPAL's Enforcement Engine. It is responsible for the internal enforcement of the privacy policies agreed upon and stated in the privacy policy sub module ,it further consists of
 1. Authorization Director:-It parses the policies and authorizes any request to access the protected resource.
 2. Resource Monitor:-Its role is to get permission of access to the protected resource and make audit of every request. This component plays an important role in the whole system as it is responsible for the maintenance of the audit logs.
 3. Policy Management System:-This system defines all the privacy policies.
 4. Audit logs:-An important part stores every access detail. It contains the entity which accessed the data,Purpose for the access, date, etc.
4. External audit Engine:-This engine implements the concept of access classes and the access map. And it is responsible for the mapping the access class to the audit logs. Also an agent platform for the Software agents is the part of the submodule. This allows for the Software agents of the third party to access the audit logs.

The Overview of the whole framework is given in Figure 4.

3.8 Hierarchy of Trusted Third Party and Their Regulators

The regulator of the trusted third party plays an important role of monitoring and regulating the trusted third party. Since the trusted third parties play such a critical role in the security and are entrusted with access to audit logs of the web services. The CPI(Controllor for Privacy Insurers) ensure that the third parties are non-biased,and in any case this is compromised,take severe action against it.

Each country has a CPI and all the trusted third parties are required to get themselves registered with the CPI in which country they have a presence. The role of CPI will be similar to the role of CCA (Controllor of Certifying Authorities) as recommended in UNICITRAL's Model IT law and implemented in Information Technology Act 2000 Chapter 6[13] .

Since the laws and rules are in place the role of the CPI can be assigned to the CCA. The control structure is given below in Figure 5.

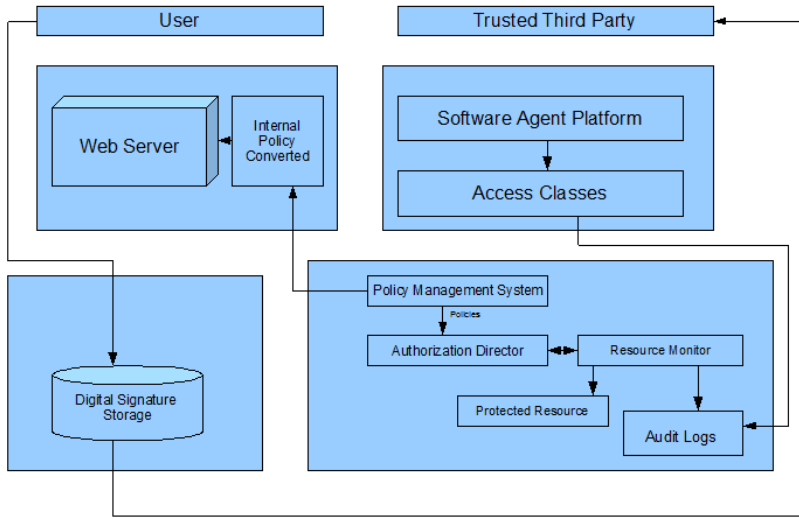


Fig. 4. Overview of the framework

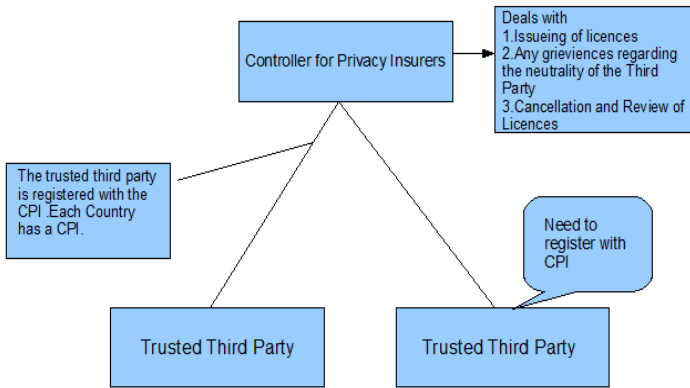


Fig. 5. Hierarchy of CPI and Trusted Third Parties

3.9 Advantages and Disadvantages of the Proposed Privacy Framework

The proposed privacy framework provides the following advantages

1. Third party evaluation-the third party plays an active role in auditing and evaluating the users PII usage.
2. Ability of the user to monitor PII use.
3. Legal Binding of transfer of PII.
4. Availability of legal remedies to address the problems of data proliferation by web services
5. Legal Framework of CPI and trusted third parties.

6. Integrity of the Web Service protected through Access Classes.

Disadvantages of the Proposed Privacy Policy

1. Complex technologies involved for example Software Agents .
2. Uses Data Intensive auditing to enforce PII's integrity.

4 Implementation Details

The whole system is implemented using Microsoft Visual Studio 2005 and the database used is Microsoft SQL Server 2005. The object oriented programming methodology is used in the implementation of the model. The policies are written using EPAL vocabulary. EPAL to P3P converter [12] is used to convert the EPAL policy to P3P and host it on the web service. The database is used to store all the data and users preferences. Access map is also in the form of a XML file. The Software Agent platform has not been implemented.

5 Future Scope and Conclusion

The proposed privacy platform solves many of the problems identified (Trusted Third Party, Legal Recognition to PII). The whole problem of privacy of PII will take even more and more significance, as the Internet penetration increases. The problems of privacy will be magnified in such a scenario, therefore the concept of third party is bound to make comebacks as it provides impartiality. The framework will also compel the web services to manage the PII in a better and secure manner.

Future scope lies in developing the Software Agent Platform and the standardization of the framework. The concept of Access Classes can be further refined and extended.

References

1. Awad, N.F., Krishnan, M.S.: The Personalization Privacy Paradox: An Empirical Evaluation of Information Transparency and the Willingness to be Profiled Online for Personalization. *MIS Quarterly* 1(30), 13–28 (2006)
2. Federal Trade Commission. Privacy online: A report to congress, <http://www.ftc.gov/reports/privacy3/>. 1998
3. Privacy Leadership Initiative. Privacy Notices Research Final Results. Conducted by Harris Interactive (December 2001), <http://www.ftc.gov/bcp/workshops/glb/supporting/harris%20>
4. Antón, A.I., Earp, J.B., Bolchini, D., He, Q., Jensen, C., Stufflebeam, W.: The Lack of Clarity in Financial Privacy Policies and the Need for Standardization. *IEEE Security & Privacy* (2004), <http://ieeexplore.ieee.org>
5. W3C: The Platform for Privacy Preferences 1.0 (P3P1.0) Specification (2002)
6. Cranor, L.F., Arjula, M., Guduru, P.: Use of a P3P User Agent by Early Adopters Workshop on Privacy In The Electronic Society. In: Proceedings of the 2002 ACM workshop on Privacy (2002)
7. Web privacy with P3P LF Cranor (2002) ISBN 81-7366-521-4

8. An Assessment of P3P and Internet Privacy, EPIC (Electronic Privacy Information Center) (June 2000)
9. Cranor, L.F., Egelman, S., Sheng, S., McDonald, A.M., Chowdhury, A.: P3P deployment on websites. In: Electronic Commerce Research and Applications. Elsevier, Amsterdam (2008)
10. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practices: Privacy-enabled management of customer data. In: Dingedine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 69–84. Springer, Heidelberg (2003)
11. (EPAL1.1) Specification. IBM Research Report, <http://www.zurich.ibm.com/security/enterprise-privacy/epal>
12. EPAL to P3P converter, <http://sourceforge.net/projects/policyconverter>
13. Sharma, V.: Information Technology Law and Practice Law of emerging Technology. Cyber Law and E-commerce (2007) ISBN-978-81-7534-619-2

Collusion-Resistant Protocol for Privacy-Preserving Distributed Association Rules Mining*

Xin-Jing Ge and Jian-Ming Zhu

School of Information, Central University of Finance and Economics
Beijing, 100081, P.R. China

Abstract. Privacy-preserving data mining (PPDM) primarily addresses the incorporation of privacy preserving concerns to data mining techniques. In this paper, we explore the problem of privacy-preserving distributed association rule mining in vertically partitioned data among multiple parties, and propose a collusion-resistant protocol of distributed association rules mining based on the threshold homomorphic encryption scheme, which can prevent effectively the collusion behaviors and conduct the computations across the parties without compromising their data privacy. In addition, the correctness, complexity and security of the collusion-resistant protocol are analyzed, and the result shows that the protocol has a reasonable efficiency and security.

Keywords: Privacy, security, association rules mining, collusion.

1 Introduction

Privacy-preserving data mining (PPDM) focuses on incorporating privacy preservation methods into data mining techniques (e.g. [3]). In this paper, we are particularly interested in the mining of association rules in a scenario where the data is vertically distributed among different parties. To mine the association rules, these parties need to collaborate with each other so that they can jointly mine the data and produce results that interest all of them. Without privacy concerns, all parties can send their data to a trusted central place to conduct the mining. However, Due to privacy law or motivated by business interests, the parties may not trust anyone and do not want to reveal her own portion of the data, although they realize that combining their data has some mutual benefit.

Privacy preserving protocols or algorithms are designed to preserve privacy even in the presence of adversarial participants that attempt to gather information about the inputs of their peers. There are, however, different levels of

* This work was supported in part by Humanities and Social Sciences Key Project of Ministry of Education of China (Grant No. 07JZD0019), the National Natural Science Foundation of China (60673162, 60970143), and the Key Project of Chinese Ministry of Education.(No109016).

adversarial behavior. Cryptographic research typically considers two types of adversaries [13]: A semi-honest adversary is a party that correctly follows the protocol specification, yet attempts to learn additional information by analyzing the messages received during the protocol execution. On the other hand, a malicious adversary may arbitrarily deviate from the protocol specification. As mentioned in previous works on privacy-preserving distributed mining [10], the participants are assumed to be semi-honest that is a practical and realistic one for distributed data mining, but malicious adversaries, for example, the collusion of parties, happen easily to gain additional benefits. Of course, we can first design a secure protocol for the semi-honest case, and then transform it into a protocol that is secure against malicious adversaries. This transformation can be done by requiring each party to use zero-knowledge proofs to prove that each step taken follows the specification of the protocol [8]. Yet, this generic approach might be rather inefficient and add considerable overhead to each step of the protocol.

In this paper, we provide a secure distributed association rules mining protocol based on the threshold additively homomorphic encryption scheme and the protocol is designed to target malicious adversaries, especially collusion-resistant. The major contributions of this paper are as follows.

- (1) We present a new solution to distributed association rules in vertically partitioned data.
- (2) We provide a secure protocol of the distributed association rules mining, which can prevent effectively the collusion of parties by employing the threshold homomorphic cryptography, and the protocol analysis shows that the protocol has a reasonable efficiency and security.

The rest of the paper is organized as follows: in Section 2 we present an overview of the related work. Subsequently, in Section 3 we present the technical preliminaries. We then present the definition of the distributed mining of association rules and the secure protocol in Section 4. Finally, we present the conclusion and future work in Section 5.

2 Related Work

Privacy-preserving data mining has received a lot of attention recently due to the increasing need to share and analyze data, and many methods were proposed to preserve the privacy of the data.

Data perturbation represents one common approach in privacy preserving data mining (PPDM), where the original (private) dataset is perturbed and the result is released for data analysis. Reference [3] first proposed randomization approaches to solve the problem of privacy-preserving in data mining. Data perturbation includes a wide variety of techniques as follows: additive, multiplicative, matrix multiplicative, k-anonymization, micro-aggregation, categorical data perturbation, data swapping, data shuffling [9]. Typically, a “privacy/accuracy” trade-off is faced. On the one hand, perturbation must not allow the original data records to be adequately recovered. On the other, it must allow “patterns” in the original data to be mined. In addition, [5] presented an approach to conduct association rule mining based on randomized response techniques.

Secure Multi-party Computation (SMC) technique is an alternative approach to achieve privacy-preserving data mining, which was introduced by [18], and has been proved that there is a secure multi-party computation solution for any polynomial function [7]. This approach, though appealing in its generality and simplicity, is highly impractical for large data sets. Based on the idea of secure multiparty computation, privacy-oriented protocols are designed for the problem of privacy-preserving collaborative data mining. Reference [16] presented the component scalar product protocol for privacy-preserving association rule mining over vertically partitioned data for the case of two parties. Reference [17] proposed an efficient and practical protocol for privacy-preserving association rule mining based on identity-based cryptography, which has an additional advantage that no public key certificate is needed.

Reference [19] proposed a homomorphic cryptographic approach to tackle collaborative association rule mining among multiple parties. The protocol we propose resemble the approach given by [19] in that we also employ a secure multiparty computation approach for preserving privacy that is based on a homomorphic cryptosystem. But we have two major differences with [19]:

- (1) In our paper, we propose the protocol of distributed association rules based on not only the homomorphic cryptographic, but also the threshold cryptography.
- (2) Our protocol is designed to target the malicious adversary, specifically collusion-resistant, which can be achieved by employing a threshold homomorphic cryptosystem, but [19] is proposed to take care of the semi-trust adversary.

3 Technical Preliminaries

3.1 Homomorphic Encryption

A cryptosystem is homomorphic [14] with respect to some operation $*$ on the message space if there is a corresponding operation $*'$ on the ciphertext space, such that $e(m_1) *' e(m_2) = e(m_1 * m_2)$. In the privacy-preserving protocol of distributed mining of association rules, we use additive homomorphism offered by [12] that is comparable with the encryption process of RSA in terms of the computation cost, while the decryption process of the additive homomorphism is faster than the decryption process of RSA.

An additively homomorphic cryptosystem has the nice property that for two plain text message m_1 and m_2 , it holds

$$e(m_1) \times e(m_2) = e(m_1 + m_2) . \tag{1}$$

where \times denotes multiplication. This essentially means that we can have the sum of two numbers without knowing what those numbers are. Moreover, because of the property of associativity, when $e(m_i) \neq 0$,

$$e(m_1) \times e(m_2) \times \dots \times e(m_n) = e(m_1 + m_2 + \dots + m_n) . \tag{2}$$

And we can easily have the following corollary:

$$e(m_1)^{m_2} = e(m_2)^{m_1} = e(m_1 \times m_2) . \tag{3}$$

3.2 Threshold Cryptography

Based on the secret sharing scheme proposed by [15], [4] presented the concept of threshold cryptography, where a secret key associated with a single public key is distributed among a group of users. Only if a predetermined number of users cooperate, can they perform some crypto-operations, such as decrypting or signing, which has drawn great attentions and enabled threshold cryptography to be widely used in many fields since proposed.

3.3 The Definition of the Association Rules Mining

The association rules mining can be defined as follows [1]: let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items, DB be a set of transactions, where each transaction T is an itemset such that $T \subseteq I$. Given an itemset $X \subseteq I$, a transaction T contains X if and only if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$ where $X \subseteq I, Y \subseteq I$ and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has support s in the transaction database DB if $s\%$ of transactions in DB contain $X \cup Y$. The association rules hold in the transaction database DB with confidence c if $c\%$ of transactions in DB that contain X also contains Y . An itemset X with k items called ***k-itemset***. The problem of mining association rules is to find all rules whose support and confidence are higher than certain user specified minimum (namely thresholds) support and confidence. With this framework, we consider mining Boolean association rules. The absence or presence of an attribute is represented as 0 or 1, so transactions are strings of 0 and 1.

4 Privacy-Preserving Distributed Mining of Association Rules in Vertically Partitioned Data

4.1 Problem Definition

In this paper, we consider the distributed mining of association rules, that is, the mining environments is distributed. Let us assume that a transaction database DB is vertical partitioned among n parties (namely P_1, P_2, \dots, P_n), where $DB = DB_1 \cup DB_2 \cup \dots \cup DB_n$, $DB_i \cap DB_j = \emptyset$, for $\forall i, j \in n, DB_i$ resides at party $P_i (1 \leq i \leq n)$. Now they want to conduct distributed association rule mining on the concatenation of their data sets, because concerned about their data privacy, no party is willing to disclose its raw data set to others, so we formulate the following privacy-preserving distributed association rule mining problem in vertically partitioned data:

problem definition

Party P_1, P_2, \dots, P_n have private data set $DB = DB_1, DB_2, \dots, DB_n$ respectively, and $DB_i \cap DB_j = \emptyset$, for $\forall i, j \in n, DB_i$. The data set $DB = DB_1, DB_2, \dots, DB_n$ forms a database DB , namely $DB = DB_1 \cup DB_2 \cup \dots \cup DB_n$, which is actually the concatenation of $DB = DB_1, DB_2, \dots, DB_n$, let N denote the total number of transactions for each data set. The n parties want to

conduct association rule mining on $DB = DB_1 \cup DB_2 \cup \dots \cup DB_n$ and to find the association rules with support and confidence being greater than the given thresholds.

During the mining of association rules, we assume all parties follow the protocol, and the object of the paper is to propose a protocol of distributed association rules mining in vertically partitioned data based on the threshold homomorphic encryption scheme, which can prevent effectively the collusion behaviors and conduct the computations across the parties without compromising their data privacy. Simultaneously, the security of the protocol refer to semantic security [6].

4.2 Association Rule Mining Algorithm

In order to learn association rules, one must compute confidence and support of a given candidate itemset, and given the values of the attributes are 1 or 0, to find out whether a particular itemset is frequent, we only count the number of records where the values for all the attributes in the itemset are 1. The following is the algorithm to find frequent itemsets:

1. $L_1 =$ large **1-itemset**
2. for ($k = 2$; $L_{k-1} \neq \emptyset$; $k++$)
3. $C_k =$ **apriori-gen** (L_{k-1})
4. for all candidates $c \in C_k$ do begin
5. Compute **c.count** (**c.count** divided by the total number of records is the support of a given itemset. We will show how to compute it in Section 4.3.)
6. end
7. $L_k = L_k \cup \{c | \mathbf{c.count} \geq \mathit{minsup}\}$
8. end
9. Return $L = \cup_k L_k$.

In step 3, the function $C_k =$ **apriori-gen** (L_{k-1}) is the following algorithm[2]:

Input: L_{i-1} // large (i-1)itemset

Output: C_i //i candidate itemset

Apriori-gen algorithms:

```

 $C_i = \emptyset$ 
for each  $I \in L_{i-1}$  do
  for each  $J \neq I \in L_{i-1}$  do
    if  $i - 2$  of the elements in  $I$  and  $J$  are equal then
       $C_k = C_k \cup I \cup J$  .
    
```

Given the counts and frequent itemsets, we can compute all association rules with $\mathit{support} \geq \mathit{minsup}$.

In the procedure of association rule mining, step 1 , 3 , 5 and 7 require sharing information. In step 3 and 7, we use merely attribute names, in step 1, to compute large **1-itemset**, each party elects her own attributes that contribute to large **1-itemset** , where only one attribute forms a **1-itemset**, there is no computation involving attributes of other parties. Therefore, data disclosure across parties is not necessary. At the same time, since the final result $L = \cup_k L_k$ is known to all parties, step 1 , 3 and 7 reveal no extra information to either party.

However, to compute **c.count** in step 5, a computation accessing attributes belonging to different parties is necessary. How to conduct these computations across parties without compromising each party’s data privacy is the challenge we are faced with. If all the attributes belong to the same party, then **c.count**, which refers to the frequency counts for candidates, can be computed by this party. If the attributes belong to different parties, they may first construct vectors for their own attributes. For example, for the some candidate itemset, party P_i have attributes a_1, a_2, \dots, a_p , then party P_i can construct vector A_i , the j th element denote $A_{ij} = \prod_{k=1}^p a_k$ in vector A_i . subsequently, they apply our secure protocols to obtain **c.count**, which will be discussed in Section 4.

4.3 Threshold Homomorphic Encryption Protocol in Vertically Partitioned Data

The fact that the collaborative parties jointly compute **c.count** without revealing their raw data to each other presents a great challenge for the case of multiple parties. Without loss of generality, assuming Party P_1 has a private vector A_1 , Party P_2 , a private vector A_2, \dots , and Party P_n , a private vector A_n . we use A_{ij} to denote the j th element in vector A_i , so the value of A_{ij} is the attribute value of the P_i in the j th transaction of the database. Given that the absence or presence of an attribute is represented as 0 or 1, the value of A_{ij} is equal to 0 or 1. For example $A_i = (1, 0, 1, 1, \dots, 0)^T$.

we develop a collusion-resistant and secure protocol to compute **c.count** as follows:

protocol

1. P_1, P_2, \dots, P_n perform the following:

(a) P_1, P_2, \dots, P_t ($1 \leq t \leq n$) jointly generate a threshold cryptographic key pair $(d(d_1, d_2, \dots d_t), e)$ of a homomorphic encryption scheme. That is, a secret key associated with a single public key is distributed among a group of parties. For simplicity and without loss of generality, let $t = n$, then only if all parties cooperate, can they decrypt the ciphertext and prevent the collusion of parties. Let $e(\cdot)$ denote encryption and $d_i(\cdot)$ denote party i decryption. Meanwhile, the threshold cryptographic key pair $(d(d_1, d_2, \dots d_n), e)$ is semantic security [6]. They also generate the number X , where X is an integer which is more than n .

(b) P_1 generates a set of random integers $R_{11}, R_{12}, \dots, R_{1N}$ and sends

$$e(A_{11} + R_{11}X), e(A_{12} + R_{12}X), \dots, e(A_{1N} + R_{1N}X)$$

to P_n ; P_2 generates a set of random integers $R_{21}, R_{22}, \dots, R_{2N}$ and sends

$$e(A_{21} + R_{21}X), e(A_{22} + R_{22}X), \dots, e(A_{2N} + R_{2N}X)$$

to P_n ; \dots ; P_{n-1} generates a set of random integers $R_{(n-1)1}, R_{(n-1)2}, \dots, R_{(n-1)N}$ and sends

$e(A_{(n-1)1} + R_{(n-1)1}X), e(A_{(n-1)2} + R_{(n-1)2}X), \dots, e(A_{(n-1)N} + R_{(n-1)N}X)$ to P_n ; P_n generates a set of random integers $R_{n1}, R_{n2}, \dots, R_{nN}$ and encrypts his private vector

$$e(A_{n1} + R_{n1}X), e(A_{n2} + R_{n2}X), \dots, e(A_{nN} + R_{nN}X).$$

(c) P_n computes:

$$\begin{aligned} E_1 &= e(A_{11} + R_{11}X) \times e(A_{21} + R_{21}X) \times \dots \times e(A_{n1} + R_{n1}X) \\ &= e(A_{11} + A_{21} + \dots + A_{n1} + (R_{11} + R_{21} + \dots + R_{n1})X) \end{aligned}$$

$$\begin{aligned} E_2 &= e(A_{12} + R_{12}X) \times e(A_{22} + R_{22}X) \times \dots \times e(A_{n2} + R_{n2}X) \\ &= e(A_{12} + A_{22} + \dots + A_{n2} + (R_{12} + R_{22} + \dots + R_{n2})X) \end{aligned}$$

.....

$$\begin{aligned} E_N &= e(A_{1N} + R_{1N}X) \times e(A_{2N} + R_{2N}X) \times \dots \times e(A_{nN} + R_{nN}X) \\ &= e(A_{1N} + A_{2N} + \dots + A_{nN} + (R_{1N} + R_{2N} + \dots + R_{nN})X) \end{aligned}$$

(d) P_n randomly permutes E_1, E_2, \dots, E_N and obtains the permuted sequence D_1, D_2, \dots, D_N .

(e) From computational balance point of view, P_n divides D_1, D_2, \dots, D_N into n parts with each part having approximately equal number of elements.

(f) P_n decrypts using himself private key d_n and sends

$$d_n(D_1), d_n(D_2), \dots, d_n(D_{N/n})$$

to P_{n-1} ; P_{n-1} decrypts

$$d_{n-1}d_n(D_1), d_{n-1}d_n(D_2), \dots, d_{n-1}d_n(D_{N/n}),$$

and send it to P_{n-2}, \dots, P_1 ; P_1 finally decrypts

$$\begin{aligned} d_1d_2 \dots d_{n-1}d_n(D_1) &= M_1, \\ d_1d_2 \dots d_{n-1}d_n(D_2) &= M_2, \\ &\dots, \\ d_1d_2 \dots d_{n-1}d_n(D_{N/n}) &= M_{N/n}. \end{aligned}$$

(g) P_n decrypts using himself private key d_n and sends

$$d_n(D_{N/n+1}), d_n(D_{N/n+2}), \dots, d_n(D_{2N/n})$$

to P_{n-1} ; P_{n-1} decrypts

$$d_{n-1}d_n(D_{N/n+1}), d_{n-1}d_n(D_{N/n+2}), \dots, d_{n-1}d_n(D_{2N/n})$$

and sends it to $P_{n-2}, \dots, P_3, P_1, P_2$; P_2 decrypts

$$\begin{aligned} d_2d_1d_3 \dots d_{n-1}d_n(D_{N/n+1}) &= M_{N/n+1}, \\ d_2d_1d_3 \dots d_{n-1}d_n(D_{N/n+2}) &= M_{N/n+2}, \\ &\dots, \\ d_2d_1d_3 \dots d_{n-1}d_n(D_{2N/n}) &= M_{2N/n}. \end{aligned}$$

(h) Continue until P_n decrypts using himself private key d_n and sends

$$d_n (D_{(n-2)N/n+1}) , d_n (D_{(n-2)N/n+2}) , d_n (D_{(n-1)N/n})$$

to P_{n-2} ; P_{n-2} decrypts and sends it to $P_{n-3}, \dots, P_2, P_1, P_{n-1}$; P_{n-1} finally decrypts

$$\begin{aligned} d_{n-1}d_1d_2 \cdots d_{n-2}d_n (D_{(n-2)N/n+1}) &= M_{(n-2)N/n+1}, \\ d_{n-1}d_1d_2 \cdots d_{n-2}d_n (D_{(n-2)N/n+2}) &= M_{(n-2)N/n+2}, \\ \dots\dots\dots, \\ d_{n-1}d_1d_2 \cdots d_{n-2}d_n (D_{(n-1)N/n}) &= M_{(n-1)N/n}. \end{aligned}$$

(i) P_n sends $D_{(n-1)N/n+1}, D_{(n-1)N/n+2}, \dots, D_N$ to P_{n-1} ; P_{n-1} decrypts and sends it to $P_{n-2}, \dots, P_2, P_1, P_n$; P_n finally decrypts

$$\begin{aligned} d_n d_1 d_2 \cdots d_{n-2} d_{n-1} (D_{(n-1)N/n+1}) &= M_{(n-1)N/n+1}, \\ d_n d_1 d_2 \cdots d_{n-2} d_{n-1} (D_{(n-1)N/n+2}) &= M_{(n-1)N/n+2}, \\ \dots\dots\dots, \\ d_n d_1 d_2 \cdots d_{n-2} d_{n-1} (D_N) &= M_N. \end{aligned}$$

2. Compute *c.count*

(a) P_1, P_2, \dots, P_n make M_1, M_2, \dots, M_N module X respectively, note that if a decrypted term M_i is equal to $n \bmod X$, it means the values of P_1, P_2, \dots, P_n are all 1, then let $m_i = 1$, otherwise $m_i = 0$. For example, if the transaction j is permuted as position i , then

$$M_i \bmod X = (A_{1j} + A_{2j} + \cdots + A_{nj} + (R_{1j} + R_{2j} + \cdots + R_{nj}) X) \bmod X.$$

Consequently, compare whether each decrypted term $M_i \bmod X$ is equal to $n \bmod X$, if yes, then let $m_i = 1$, otherwise $m_i = 0$.

(b) P_1 computes $c_1 = \sum_{i=1}^{N/n} m_i$, P_2 computes $c_2 = \sum_{i=N/n+1}^{2N/n} m_i$, \dots , P_n computes $c_n = \sum_{i=(n-1)N/n+1}^N m_i$.

(c) all parties $P_i (1 \leq i \leq n)$ encrypt $e(c_i)$ and send it to P_n .

(d) P_n computes

$$e(c_1) \times e(c_2) \times \cdots \times e(c_n) = e(c_1 + c_2 + \cdots + c_n),$$

then decrypts

$$d_n (e(c_1 + c_2 + \cdots + c_n))$$

and sends it to P_{n-1} ; P_{n-1} decrypts

$$d_{n-1}d_n (e(c_1 + c_2 + \cdots + c_n))$$

and sends it to P_{n-2}, \dots, P_1 decrypts

$$d_1 \cdots d_{n-1}d_n (e(c_1 + c_2 + \cdots + c_n)) = c_1 + c_2 + \cdots + c_n = \mathbf{c.count}.$$

4.4 Analysis of the Secure Distributed Association Rules Mining Protocol

Correctness Analysis. Assume all of the parties follow the protocol, in which the threshold cryptographic system is a additively homomorphic cryptosystem, which enable us to get

$$\begin{aligned} E_i &= e(A_{1i} + R_{1i}X) \times e(A_{2i} + R_{2i}X) \times \cdots \times e(A_{ni} + R_{ni}X) \\ &= e(A_{1i} + A_{2i} + \cdots + A_{ni} + (R_{1i} + R_{2i} + \cdots + R_{ni})X) \end{aligned}$$

where $1 \leq i \leq N$. And given $X > n$, so

$$M_i \bmod X = (A_{1j} + A_{2j} + \cdots + A_{nj} + (R_{1j} + R_{2j} + \cdots + R_{nj})X) \bmod X.$$

If $A_{1j} + A_{2j} + \cdots + A_{nj}$ are all equal to 1, this means the transaction has the whole attributes and supports the association rule, we let $m_i = 1$. Otherwise, if some attributes of are not equal to 1, this means the transaction has not the whole attributes and does not support the association rules, we let $m_i = 0$, to compute the number of transactions which support the association rule, we only count the number of $m_i = 1$, then $c_1 + c_2 + \cdots + c_n = \mathbf{c.count}$.

Meanwhile, in the protocol, P_n permutes E_i ($1 \leq i \leq N$) before sending them to other parties, permutation does not affect $\mathbf{c.count}$, and summation is not affected by a permutation. Therefore, the final $\mathbf{c.count}$ is correct.

Complexity Analysis. The bit-wise communication cost of this protocol is upper bounded by

$2\alpha[(n-1)N+n]$, where α is the number of bits for each encrypted element. It consist of (1) the maximum cost of $(n-1)N$ from step 1(b); (2) the maximum cost of $(n-1)N$ from step 1(f)-1(i); (3) the maximum cost of $2n$ from step 2(c) and 2(d).

The following contributes to the computational cost: (1) the generation of a threshold cryptographic key pair, the integer X and nN random integers; (2) the total number of $nN+n$ encryptions; (3) the total number of $(n-1)(N+1)$ multiplications; (4) the generation of permutation function; (5) the total number of N permutations; (6) the total number of $(n+1)N$ decryptions; (7) the total number of N modulo operations; (8) the total number of $(n+1)N$ additions; (9) dividing N numbers into n parts.

The communication cost and computational cost of our protocol are approximately equal to those of [19], except for the increased complexity in the decryptions for the prevention of collusion.

Security Analysis. Given that P_n obtains all the encrypted terms from other parties and the cryptographic system is a semantic security, the ciphertext does not leak any useful information about the plaintext and P_n can not get other useful information of the plaintext from the ciphertext. Meanwhile, since the cryptographic system is a threshold cryptosystem, those parties will not able to decrypt and get the plaintext unless they cooperate. That is, P_n will not

have access to the original values of other parties without cooperating with those parties. As a result, the collusion behaviors can be prevented effectively. In our protocol P_1, P_2, \dots, P_n jointly generate a threshold cryptographic key pair $(d(d_1, d_2, \dots, d_n), e)$ of a homomorphic encryption scheme, which means the protocol is secure under the condition of the number of the collusion parties is less than n . Generally, given P_1, P_2, \dots, P_t ($1 \leq t \leq n$) jointly generate a threshold cryptographic key pair $(d(d_1, d_2, \dots, d_n), e)$ of a homomorphic encryption scheme, which means the protocol is secure under the condition of the number of the collusion parties is less than t .

Meanwhile, each party of P_1, P_2, \dots, P_{n-1} obtains some plaintexts of all D_i . Since D_i are in permuted form and those $n - 1$ parties don't know the permutation function, so they cannot know which transaction support the association rule. And each party only knows a part of transactions supporting the association rules, which lead to trivial benefit for them.

5 Conclusion

This paper concerns itself with the issue of privacy-preserving distributed association rule mining. In particular, we focus on how multiple parties can conduct distributed association rule mining in their joint database of the vertically partitioned private data, in which we propose a collusion-resistant protocol for privacy-preserving distributed association rules mining based on the threshold homomorphic encryption scheme. In-depth analysis of correctness, complexity and security about our protocols is given, and the result shows that the protocol has a reasonable efficiency and security. Our technique can be further explored and applied to other data mining computations, such as decision tree classification.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD Conference on Management of Data, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, pp. 12–15 (1994)
3. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 439–450. ACM Press, New York (2000)
4. Desmedt, Y.G.: Threshold cryptography. European Trans. on Telecommunications 5(4), 449–457 (1994)
5. Evmimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, pp. 217–228 (2002)
6. Goldreich, O.: Foundations of cryptography, Class notes, Technion University (Spring 1989)

7. Goldreich, O.: Secure multi-party computation (1998), <http://www.wisdom.weizmann.ac.il/>
8. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM* 38, 691–729 (1991)
9. Lin, K., Giannella, C., Kargupta, H.: A survey of attack techniques on privacy-preserving data perturbation. In: *Privacy-Preserving Data Mining: Models and Algorithms*, pp. 275–299 (2008)
10. Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. *Journal of Cryptology* 15(3), 177–206 (2002)
11. Luby, M.: *Pseudorandomness and Cryptographic Applications*. Princeton University Press, Princeton (1996)
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) *EUROCRYPT 1999*. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
13. Pinkas, B.: Cryptographic techniques for privacy preserving Data mining. *SIGKDD Explorations* 4(2), 12–19 (2002)
14. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: De Millo, R.A., et al. (eds.) *Foundations of Secure Computation*, pp. 169–179. Academic Press, London (1978)
15. Shamir, A.: How to share a secret. *Commun. ACM* 22, 612–613 (1979)
16. Vaidya, J., Clifton, C.W.: Privacy preserving association rule mining in vertically partitioned data. In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Canada (2002)
17. Wu, F., Liu, J.q., Zhong, Sh.: An efficient protocol for private and accurate mining of support counts 30(1), 80–86 (January 2009)
18. Yao, A.C.: Protocols for secure computations. In: *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*. IEEE Press, New York (1982)
19. Zhan, J., Matwin, S., Chang, L.W.: Privacy-preserving collaborative association rule mining. *Journal of Network and Computer Applications* 30(3), 1216–1227 (2007)

GUC-Secure Join Operator in Distributed Relational Database

Yuan Tian and Hao Zhang

Software School, Dalian University of Technology,
Dalian, Liaoning, 116620, China
tianyuan_ca@sina.com, hzhang@dlut.edu.cn

Abstract. Privacy-preserving SQL computation in distributed relational database is one of important applications of secure multiparty computation. In contrast with comparatively more works on privacy-preserving data-query in database, only few works deal with provably-secure privacy-preserving data manipulations, among which the join operator is the most powerful in generating new data (relation). This paper proposes a very general cryptographic protocol framework for secure 2-party join computation based on anonymous IBE (identity-based encryption) scheme and its user private-keys blind generation techniques. This construction is provably GUC (generalized universally composable) secure in standard model with acceptable efficiency.

Keywords: Secure Multiparty Computation, Distributed Relational Database, Anonymous Identity-based Encryption, Generalized Universally Composable Security.

1 Introduction

Relational database (RDBMS) is one of the most widely deployed and mission-critical information systems. In addition, distributed relational database technology has become one of the major recent developments in this area [15]. In distributed RDBMS, unlike a single centralized RDBMS server, information is distributed among multiple sites and some pieces of this information are highly private for its owners. Therefore a natural secure multiparty computation problem arises in this setting: how to generate the useful information while keeping all participating sites' data privacy?

In RDBMS information is modeled as relations or tables. Such highly structured information is processed by the powerful relational algebra operators, e.g., (constrained) selection, projection, product and join operators, which are applied to one or multiple tables to generate new table [7]. Set operators, e.g., union, intersection, insert, etc., are also regarded as extended relational algebra operators. All these have been formally standardized as the RDBMS programming language SQL (Structured Query Language) which is widely used even in distributed RDBMS systems.

Motivations. So far lots of works have been done to develop solutions to secure SQL computations in distributed RDBMS and the most majority comes from database rather than cryptography community. However, from the cryptography theorist’s perspective, only few of them are reliable: most works, although highly heuristic, do not have precise security analysis or even without precise security definitions. As a (unfortunate) result, most of them are actually either insecure under the standard security definition currently accepted by cryptographers or unable-to-be-proved. In particular, since the SQL-operator is always called by high-level programs, its security must be preserved no matter how it is called and combined with any other (maybe malicious) running contexts. In other words, secure solutions to SQL-operators must be provably UC or GUC secure [4,5]. But with our knowledge none of such solution exists in this area. For this reason, we won’t further review these works from database community (interested readers can refer the survey [9] and its references) and only focus on provably secure solutions.

Among the provably secure solutions to distributed data processing (not limited to distributed RDBMS), most of them are focused on data query (i.e., read-only) rather than data manipulation operators. Although data query is one of the most important and frequently-used operations in distributed systems, lack of secure solutions to data manipulation operators will become a bottleneck in deploying highly security-critical applications. Roughly speaking, secure data query solutions are applied on encrypted data (e.g., [1,2,11,16,17], many approaches are not limited to relational data model) to extract the desired information via smartly designed trapdoor mechanisms. All these solutions can provide techniques for constructing secure SQL-query operators such as the most frequently-used constrained selection operators. Generally speaking, secure data manipulation is more technically involved than data query schemes and so far there are indeed few works on this problem, among which [10,12,13] provide the provably secure solutions. [10,13] can be also regarded as the secure solution to SQL set operators, but none of them reaches Canetti’s UC/GUC-security [4,5] ([10] works in two relaxed adversary models to achieve security of “half-simulatability” and “full-simulatability against covert adversaries”; [13] is provably secure in terms of full-simulatability which is still strictly weaker than UC/GUC-security). With our knowledge, there’re no satisfactory secure solutions to important relational data manipulation operators such as join and its variants.

In this paper we focus on secure join computation. Join is a powerful SQL operator to generate a new table from existing tables in RDBMS. For example, given two tables, *Incom* with attributes “*customer_id*” and “*income*”, *Debt* with attributes “*customer_id*” and “*debt*”, then the joint operator $Join(customer_id : Incom, Debt)$ produces a new table with attributes “*customer_id*”, “*income*” and “*debt*” which entries are derived from the product of the entries in table *Incom* and *Debt* which have the same values in *customer_id* field. For instance, suppose *Incom* has entries $\{(c1, 2500), (\boxed{c2}, 3000), (\boxed{c5}, 1010), (c6, 2000)\}$, *Debt* has entries $\{(\boxed{c2}, 19000), (c4, 7000), (\boxed{c5}, 88), (c7, 100)\}$, then the above join operator outputs the entries $\{(c2, 3000, 19000), (c5, 1010, 88)\}$. Join computation with equality constraints on multiple common attributes is also useful in practice.

Furthermore, if the table *Incom* and *Debt* are stored at different sites and the information about those uncommon customers must be kept secret to each other site, this join operator must be securely implemented in consistency with the cryptographic multiparty computation definition, i.e., each site knows nothing about the other table beyond the join operator's output.

Contributions. In this paper we construct a GUC-secure [4,5] protocol for join computation in standard model with acceptable efficiency. Like many theoretical works, we focus on the 2-party case. Our approach is very general based-on the anonymous IBE scheme and it's user private-key's blind generation techniques (i.e., to generate the correct user private-key $usk(a) = UKG(msk, a)$ for a user without leaking the user-id a to the key-generator). The protocol's high-level description is simple: let $\Pi = (Setup, UKG, E, D)$ be an IBE scheme both anonymous and data-private, M_0 be some publicly-known plaintext. Site P_1 owns the table X_1 with attributes w and x and site P_2 owns table X_2 with attributes w and y . The goal is securely computing $Join(w : X_1, X_2)$ and outputting at P_2 ¹. Suppose X_1 is filled with entries $\{(w_1, x_1), (\boxed{w_2}, x_2), (w_3, x_3), (\boxed{w_4}, x_4)\}$ and X_2 is filled with $\{(\boxed{w_2}, y_2), (\boxed{w_4}, y_4), (w_5, y_5), (w_7, y_7)\}$ where w_i 's, x_i 's and y_i 's are values of the attribute w , x and y respectively. P_1 generates IBE's master public/secret-key (mpk, msk) , sends mpk and all $\xi_i = E(mpk, w_i, x_i || M_0)$ ($i = 1, 2, 3, 4$) to P_2 . When P_2 tries to decipher each ξ_i by private-keys $usk(w_2)$, $usk(w_4)$, $usk(w_5)$ and $usk(w_7)$ (obtained via Π 's user private-keys blind generation protocol), only $usk(w_2)$ and $usk(w_4)$ can succeed in obtaining the plaintext with a suffix M_0 . As a result, P_2 gets X_1 's entries $\{(w_2, x_2), (w_4, x_4)\}$ and can now get the result of $Join(w : X_1, X_2)$, i.e., $\{(w_2, x_2, y_2), (w_4, x_4, y_4)\}$, by a local join computation. Note that Π 's anonymity and data-privacy prevents P_2 from knowing anything about X_1 beyond $\{(w_2, x_2), (w_4, x_4)\}$ while the private-key generation protocol's blindness prevents P_1 from knowing anything about X_2 .

This protocol's incorrectness probability is not 0, but by choosing M_0 lengthy enough, e.g., 128-bit, its incorrectness probability can be negligible in practice.

To be GUC-secure the formal construction is more involved (section 3). It is constant-round in communications and linear-size in message-complexity. In computation-complexity, one party is $O(N_1 + N_2)$ and the other is $O(N_1 N_2)$ encryptions/decryptions where N_1, N_2 are each party's private table's cardinality. Note that $O(N_1 N_2)$ is also local join operator's computation complexity [7], i.e., neglecting a constant factor our construction's efficiency is asymptotically the same as that of conventional join operator.

The formal construction is also well-modularized, only executing few zero-knowledge proofs of knowledge which can be efficiently instantiated. Most importantly and distinctively, our construction is provably GUC-secure against malicious adversaries assuming static corruptions in the ACRS(augmented common reference string) model [5]. For this goal we introduce a notion of identity-augmented non-malleable zero-knowledge proofs of knowledge which may be of independent values.

¹ In most cases a distributed SQL-transaction outputs its final result at some particular site.

2 Notations, Definitions and Tools

P.P.T. means “probabilistic polynomial-time”, $x||y$ means string x and y in concatenation, $|x|$ means string x ’s size (in bits) and $|X|$ (X is a set) means X ’s cardinality, $x \leftarrow^{\$} X$ means randomly selecting x from the domain X . k denotes the complexity parameter. \approx^{PPT} stands for *computational indistinguishability* and \approx for *perfect indistinguishability*.

2.1 Secure Join Computation and Its GUC Security

Briefly speaking, GUC-security means that any adversary attacking the real-world protocol can be efficiently simulated by an adversary attacking the ideal-world functionality, both have the outputs indistinguishable by the (malicious) environment. For space limitations, we assume the reader’s familiarity with the whole theory in [4-6] and only provide necessary descriptions with respect to the secure join computation problem here.

Similar to many theoretical works, we focus on the 2-party scenario. Furthermore, in this paper we only consider the horizontal distribution setting in which each table is at a site as a whole, none table is separated among different sites. Let X_1 and X_2 denote the tables with attributes $w, u^{(1)}$ and $w, u^{(2)}$ respectively, w is the common attribute and for equijoin we consider only single such common attribute without loss of generality (in case of multiple common attributes, e.g., v and w , we simply consider a imaginary single attribute $v||w$ which values are just the concatenation of values of v and w). The ideal cryptographic functionality to perform equijoin computation on X_1 and X_2 with equality constraint on w is defined as

$$F_{Join} : (X_1, X_2) \rightarrow (|w|_2, |X_1||Join(w : X_1, X_2))$$

where $|w|_2$ means the number of different values of w in X_2 and $Join(w : X_1, X_2)$ means the result of ideal relational join computation. More precisely, let P_1^*, P_2^* be parties in ideal model with private tables X_1 and X_2 respectively, $N_1 = |X_1|$, $N_2 = |X_2|$, S be the adversary in ideal model, the ideal model works as follows:

On receiving message $(sid, \text{“input”}, P_1^*, X_1)$ from P_1^* , F_{Join} records X_1 and sends message $(sid, \text{“input”}, N_1)$ to P_2^* and S ; On receiving message $(sid, \text{“input”}, P_2^*, X_2)$ from P_2^* , F_{Join} records X_2 and sends $(sid, \text{“input”}, |w|_2)$ to P_1^* and S .

On receiving message $(sid, \text{“equijoin”}, P_2^*)$ from P_2^* , F_{Join} responses P_2^* with message $(sid, \text{“equijoin”}, Join(w : X_1, X_2))$.

At last P_1^* outputs $|w|_2$, P_2^* outputs $N_1||Join(w : X_1, X_2)$.

Let ψ be the real-world protocol, each party P_i of ψ corresponds to an ideal-world party P_i^* . A is the real-world adversary attacking ψ , Z is the environment in which the real protocol/ideal functionality executes. According to [4-5], Z is a P.P.T. machine modeling all malicious behaviors against the protocol’s execution. Z is empowered to provide inputs to parties and interacts with A and S ,

e.g., Z gives special inputs or instructions to A/S , collects outputs from A/S to make some analysis, etc. In UC theory [4], Z cannot access parties' shared functionality (such shared functionality is specified in specific protocol) while in the improved GUC theory [5] Z is enhanced to do this, i.e., to provide inputs to and get outputs from the shared functionality. As a result, in GUC theory Z is strictly stronger and more realistic than in UC theory.

Let $output_Z(\psi, A)$ denote the outputs (as a joint stochastic variable) from ψ 's parties P_1, P_2 under Z and A , $output_Z(F_{INT}, S)$ denote the similar thing under Z and S . During the real/ideal protocol's execution, Z (as an active distinguisher) interacts with A/S and raises its final output, w.l.o.g., 0 or 1. Such output is denoted as $Z(output_Z(\psi, A), u)$ and $Z(output_Z(F_{INT}, S), u)$ respectively, where u is the auxiliary information.

Definition 2.1 (GUC security [5]). If for any P.P.T. adversary A in real-world, there exists a P.P.T. adversary S (called A 's simulator) in ideal-world, both corrupt the same set of parties, such that for any environment Z the function $|Pr[Z(output_Z(\psi, A), u) = 1] - Pr[Z(output_Z(F_{Join}, S), u) = 1]|$ is negligible in complexity parameter k (hereafter denote this fact as $output_Z(\psi, A) \approx^{PPT} output_Z(F_{Join}, S)$), then we define that ψ GUC-emulates F_{Join} or say ψ is GUC-secure, denoted as $\psi \rightarrow^{GUC} F_{Join}$.

The most significant property of GUC-security is the universal composition theorem [4,5].

2.2 IBE Scheme, Its Anonymity and Blind User-Private Key Generation Functionality

In addition to data-privacy, anonymity (key-privacy) is another valuable property for public-key encryption schemes [2]. An IBE scheme $\Pi = (Setup, UKG, E, D)$ is a group of P.P.T. algorithms, where $Setup$ takes the complexity parameter k as input to generate master public/secret-key pair (mpk, msk) , UKG takes msk and user's id a as input to generate a 's user private-key $usk(a)$; E takes (mpk, a, M) as input where M is the message plaintext to generate ciphertext y , D takes $(mpk, usk(a), y)$ as input to do decryption. Altogether these algorithms satisfy the consistency property: for any k, a and M

$$Pr[(mpk, msk) \leftarrow Setup(k); usk(a) \leftarrow UKG(msk, a); y \leftarrow E(mpk, a, M) : D(mpk, usk(a), y) = M] = 1$$

Definition 2.2 (IBE Scheme's Chosen Plaintext Anonymity [1]). Given an IBE scheme $\Pi = (Setup, UKG, E, D)$, for any P.P.T. attacker $A = (A_1, A_2)$ consider the following experiment $Exp_{\Pi, A}^{ANO-CPA}(k)$:

$$\begin{aligned} &(mpk, msk) \leftarrow Setup(k); \\ &(M^*, a_0^*, a_1^*, St) \leftarrow A_1^{UKG(msk, \cdot)}(mpk), a_0^* \neq a_1^*; \\ &b \leftarrow_{\$} \{0, 1\}; \\ &y^* \leftarrow E(mpk, a_b^*, M^*); \\ &d \leftarrow A_2^{UKG(msk, \cdot)}(St, y^*); \end{aligned}$$

output($d \oplus b$);

A is constrained not to query its oracle $UKG(msk, \cdot)$ with a_0^* and a_1^* . Define $Adv_{\Pi, A}^{ANO-CPA}$ as $|2Pr[Exp_{\Pi, A}^{ANO-CPA}(k) = 1] - 1|$. If $Adv_{\Pi, A}^{ANO-CPA}$ is negligible in k for any P.P.T. A then Π is called *anonymous against chosen plaintext attack* ($ANO-CPA$ for short). In the above, if M^* , a_0^* , a_1^* are generated independent of mpk then Π is called *selective ANO-CPA*.

Denote $\max_{A \in P.P.T.} Adv_{\Pi, A}^{ANO-CPA}(k)$ as $Adv_{\Pi}^{ANO-CPA}(k)$ or $Adv_{\Pi}^{ANO-CPA}(t, q)$ where t is the adversary's maximum time-complexity and q is the maximum number of queries for the UKG-oracle.

Now we present the ideal functionality $F_{Blind-UKG}^{\Pi}$ for an IBE scheme Π 's user private-key blind generation (note: even IBE scheme is not anonymous such functionality still makes sense. However, in this paper only anonymous IBE's such protocol is needed). In the ideal model, one party generates (just one time) Π 's master public/secret-key pair (mpk, msk) and submits it to $F_{Blind-UKG}^{\Pi}$; $F_{Blind-UKG}^{\Pi}$ generates $usk(a) = UKG(msk, a)$ for another party who submits its private input a (this computation can take place any times and each time for a new a), revealing nothing about a to the party who provides (mpk, msk) except how many private-keys are generated. Formally, let S be the ideal adversary, P_1^* , P_2^* the ideal party, sid and $ssid$ the session-id and sub-session-id respectively, the ideal model works as follows:

P_1^* selects randomness ρ and computes $(mpk, msk) \leftarrow Setup(\rho)$, sends the message $(sid, mpk || msk || \rho)$ to $F_{Blind-UKG}^{\Pi}$; $F_{Blind-UKG}^{\Pi}$ sends message (sid, mpk) to P_2^* and S ;

On receiving a message $(sid || ssid, a)$ from P_2^* ($ssid$ and a are fresh everytime), in response $F_{Blind-UKG}^{\Pi}$ computes $usk(a) \leftarrow UKG(msk, a)$, sends the message $(sid || ssid, usk(a))$ to P_2^* and the message $(sid || ssid, n)$ to P_1^* and S , where n is initialized to be 0 and increased by 1 everytime the computation takes place.

At last, P_1^* outputs its last n , P_2^* outputs all its obtained $usk(a)$'s.

2.3 (Identity-Augmented) Non-malleable Zero-Knowledge Proofs of Knowledge

This subsection presents the concept of zero-knowledge proofs of knowledge following [8,14] with slight symbolic modifications. Let L be a NP language, R is its associated P-class binary relation. i.e., $x \in L$ iff there exists w such that $R(x, w) = 1$. Let A, B be two machines, then $A(x; B)_{[\sigma]}$ represents A 's output due to its interactions with B under a public common input x and common reference string (c.r.s.) σ , $tr_{A, B}(x)_{[\sigma]}$ represents the transcripts due to interactions between A and B under a common input x and c.r.s. σ . When we emphasize A 's private input, say y , we also use the expression $A_y(x; B)_{[\sigma]}$ and $tr_{A(y), B}(x)_{[\sigma]}$ respectively. Let $A = (A_1, A_2)$, B and C be machines where A_1 can coordinate with A_2 by transferring status information to it, then $(\langle B, A_1 \rangle, \langle A_2, C \rangle)$ represents the interaction between A_1 and B , (maybe concurrently) A_2 and C . Due

to such interactions, let tr be the transcripts between A_2 and C , u be the final output from A_2 and v be the final output from C , then $(\langle B, A_1 \rangle, \langle A_2, C \rangle)$'s output is denoted as (u, tr, v) .

Two transcripts tr_1 and tr_2 are *matched* each other, if tr_1 and tr_2 are the same message sequence (consisted of the same messages in the same order) and the only difference is that any corresponding messages are in the opposite directions.

Let A be a machine, the symbol \boxed{A} represents such a machine which accepts two kinds of instructions: the first one is in the form of (“start”, i, x, w) and \boxed{A} in response starts a new instance of A , associates it with a unique name i and provides it with public input x and private input w ; the second is in form of (“message”, i, m) and \boxed{A} in response sends message m to instance A_i and then returns A_i 's response to m .

Definition 2.3 (Zero-Knowledge Proof and Non-malleable Zero-Knowledge Proof Protocol [14].) $ZPoK_R = (D_{crs}, P, V, Sim = (Sim_1, Sim_2))$ is a group of P.P.T. algorithms where D_{crs} takes k (complexity parameter) as input to generate c.r.s. σ ; P (called *prover*) takes (σ, x, w) as input where $R(x, w) = 1$ to generate a proof π ; V (called *verifier*) takes (σ, x) as input to generate 0 or 1; $Sim_1(k)$ generates (σ, s) , Sim_2 takes $x \in L$ and (σ, s) as input to generate the simulation. All algorithms except D_{crs} and Sim_1 take the c.r.s. σ as one of their input, so σ is no longer explicitly included in all the following expressions unless for emphasis. Now $ZPoK_R$ is defined as a *zero-knowledge proof protocol for relation R* , if the following properties are all satisfied:

- (1) For any $x \in L$ and $\sigma \leftarrow D_{crs}$, it's always true that $Pr[V(x; P)_{[\sigma]} = 1] = 1$;
- (2) For any P.P.T. algorithm A , $x \notin L$ and $\sigma \leftarrow D_{crs}$, it's always true that $Pr[V(x; A)_{[\sigma]} = 1] = 0$;
- (3) For any P.P.T. algorithm A which outputs 0 or 1, let ε be empty string, the function

$$|Pr[\sigma \leftarrow D_{crs}; b \leftarrow A(\varepsilon; \boxed{P})_{[\sigma]} : b = 1] - Pr[(\sigma, s) \leftarrow Sim_1(k); b \leftarrow A(\varepsilon; \boxed{Sim_2(s)})_{[\sigma]} : b = 1]|$$

is always negligible in k , where we emphasize the fact by symbol $Sim_2(s)$ that all Sim_2 instances have the same s as one of their inputs.

The *non-malleable zero-knowledge proof protocol* for relation R is defined as $NMZPoK_R = (D_{crs}, P, V, Sim = (Sim_1, Sim_2), Ext = (Ext_1, Ext_2))$ where (D_{crs}, P, V, Sim) is a zero-knowledge proof protocol for relation R as above, P.P.T. algorithm $Ext_1(k)$ generates (σ, s, τ) and the interactive P.P.T. machine Ext_2 (called witness extractor) takes (σ, τ) and protocol's transcripts as its input and extracts w , and all the following properties hold:

- (4) The distribution of the first output of Sim_1 is identical to that of Ext_1 ;
- (5) For any τ , the distribution of the output of V is identical to that of Ext_2 's

² Strictly this protocol should be called “zero-knowledge argument”, however, such difference is not essential in this paper so we harmlessly abuse the terminology.

restricted output which does not include the extracted $value(w)$;

(6) There exists a negligible function $\eta(k)$ (named as knowledge-error function) such that for any P.P.T. algorithm $A = (A_1, A_2)$ it's true that

$$\begin{aligned}
 & Pr[(\sigma, s, \tau) \leftarrow Ext_1(k); (x, tr, (b, w)) \leftarrow (\langle \boxed{Sim_2(s)}, A_1 \rangle, \langle A_2, Ext_2(\tau) \rangle)_{[\sigma]} : \\
 & b = 1 \wedge R(x, w) = 1 \wedge tr \text{ doesn't match any transcript generated by } \boxed{Sim_2(s)} \\
 & > Pr[(\sigma, s) \leftarrow Sim_1(k); (x, tr, b) \leftarrow (\langle \boxed{Sim_2(s)}, A_1 \rangle, \langle A_2, V \rangle)_{[\sigma]} : \\
 & b = 1 \wedge tr \text{ doesn't match any transcript generated by } \boxed{Sim_2(s)}] - \eta(k).
 \end{aligned}$$

It's easy to see that $NMZPoK_R$ is a zero-knowledge proof of knowledge. [8,14] developed an efficient method to derive non-malleable zero-knowledge proof protocols based-on simulation-sound tag-based commitment schemes and the so-called Ω -protocols (proposed in [14]). In order to achieve GUC-security in our construction, we need to further enhance $NMZPoK$ to the concept of identity-augmented non-malleable zero-knowledge proof protocol (IA- $NMZPoK$) as follows.

Definition 2.4 (IA- $NMZPoK$ Protocol for Relation R). The IA- $NMZPoK$ Protocol for relation R , $IA-NMZPoK_R = (D, Setup, UKG, P, V, Sim = (Sim_1, Sim_2), Ext = (Ext_1, Ext_2))$ is a group of P.P.T. algorithms. $Setup(k)$ generates master public/secret key-pair (mpk, msk) , $UKG(msk, id)$ generates id 's private-key $usk(id)$ where $id \in \{P, V\}$ (the prover's and verifier's identity). Sim_1 takes $usk(V)$ as input, Ext_1 takes $usk(P)$ as input. All algorithms except $Setup$ take (mpk, σ) as one of its inputs (so it no longer explicitly appears). The protocol has the same properties as R 's $NMZPoK$ protocol in definition 2.3.

Note that by this definition an IA- $NMZPoK$ protocol works in ACRS model [5] which ACRS is its mpk . In addition, only the corrupt verifier can run Sim (Sim_1 taking $usk(V)$ as input) and only the corrupt prover can run Ext (Ext_1 taking $usk(P)$ as input). This is exactly what is required in the ACRS model. Given a relation R , a general and efficient construction of IA- $NMZPoK$ protocol for R is presented in Appendix D of this paper's full version(eprint.iacr.org/2009/204).

2.4 Commitment Scheme

We need the non-interactive identity-based trapdoor commitment scheme [5] (IBTC for short) as another important tool in our construction.

Definition 2.5 (IBTC scheme [5]). Let k be complexity parameter, the non-interactive identity-based trapdoor commitment scheme $IBTC = (D, Setup, UKG, Cmt, Vf, FakeCmt, FakeDmt)$ is a group of P.P.T. algorithms, where $D(k)$ generates id , $Setup(k)$ generates master public/secret-key pair (mpk, msk) , $UKG(msk, id)$ generates id 's user private-key $usk(id)$, $Cmt(mpk, id, M)$ generates message M 's commitment/decommitment pair (cmt, dmt) , $Vf(mpk, id, M, cmt, dmt)$ outputs 0 or 1, verifying whether cmt is M 's commitment with respect to id . These algorithms are consistant, i.e., for any M :

$$\begin{aligned} &Pr[(mpk, msk) \leftarrow Setup(k); (cmt, dmt) \leftarrow Cmt(mpk, id, M) : \\ &Vf(mpk, id, M, cmt, dmt) = 1] = 1 \end{aligned}$$

$FakeCmt(mpk, id, usk(id))$ generates $(\overline{cmt}, \lambda)$, $FakeDmt(mpk, M, \lambda, \overline{cmt})$ generates \bar{d} (w.l.o.g. λ contains $id||usk(id)$ as one of its components so $FakeDmt$ doesn't explicitly take id and $usk(id)$ as its input). A secure IBTC scheme has the following properties:

- (1) Hiding: for any id and $M_0, M_1, (cmt_i, dmt_i) \leftarrow Cmt(mpk, id, M_i), i = 0, 1$, then $cmt_0 \approx^{P.P.T.} cmt_1$;
- (2) Binding: for any P.P.T. algorithm A , the function $Adv_{IBTC, A}^{binding}(k) \equiv Pr[(mpk, msk) \leftarrow Setup(k); (id^*, cmt^*, M_0^*, d_0^*, M_1^*, d_1^*) \leftarrow A^{UKG(msk, \cdot)}(mpk) : A$ doesn't query oracle- $U(msk, \cdot)$ with $id^* \wedge M_0^* \neq M_1^* \wedge Vf(mpk, id^*, M_0^*, cmt^*, d_0^*) = Vf(mpk, id^*, M_1^*, cmt^*, d_1^*) = 1]$ is always negligible in k .
- (3) Equivocability: For any P.P.T. algorithm $A = (A_1, A_2)$ the following experiment always has $|Pr[b^* = b] - 1/2|$ upper-bounded by a negligible function in k :

$$\begin{aligned} &(mpk, msk) \leftarrow Setup(k); \\ &(St, id^*, M^*) \leftarrow A_1(mpk, msk); \\ &usk(id^*) \leftarrow UKG(msk, id^*); (\overline{cmt}, \lambda) \leftarrow FakeCmt(mpk, id^*, usk(id^*)); \\ &d_1 \leftarrow FakeDmt(mpk, M^*, \lambda, \overline{cmt}); d_0 \leftarrow_{\$} \{0, 1\}^{|d_1|}; \\ &b \leftarrow_{\$} \{0, 1\}; \\ &b^* \leftarrow A_2(St, d_b); \end{aligned}$$

Note that equivocability implies $Pr[Vf(mpk, id^*, M^*, \overline{cmt}, d_1^*) = 1] > 1 - \gamma(k)$ where $\gamma(k)$ is a negligible function in k . [5] presented an efficient IBTC construction and its security proof.

3 General Construction

Now we present the formal construction of the real-world private equijoin protocol Ψ . P_1 and P_2 denote two real-world parties with private tables X_1 and X_2 , each with attributes w, x and w, y respectively (more generally x is the vector of all attributes in X_1 other than w , similar for y , but this is immaterial in our approach), $X_1 = \{(u_1, x_1), \dots, (u_N, x_N)\}, X_2 = \{(v_1, y_1), \dots, (v_N, y_N)\}$ where u_i 's, v_j 's are values of w , x_i 's are values of x and y_j 's are values of y . $\Pi = (ESetup, UKG, E, D)$ is a id-selective ANO-CPA anonymous and id-selective IND-CPA data-private IBE scheme, $\Delta_{Blind-UKG}^{\Pi}$ is the real-world protocol for Π 's user private-keys blind generation. In addition, we suppose a predefined bijective coding function H mapping strings or values (e.g., x_i, y_j) to Π 's plaintexts, but for simplicity we always write $E(mpk, a, x||y)$ instead of $E(mpk, a, H(x||y))$. IA-NMZPoK($w : R(x, w) = 1$) denotes an IA-NMZPoK protocol for relation R where w is x 's witness. $TC = (D, TSetup, UKG, Cmt, Vf, FakeCmt, FakeDmt)$ is an IBTC scheme. M_0 is a (fixed) public common string and $|M_0| = poly(k)$. Ψ 's ACRS is $mpk_{TC}||mpk_{\Delta}||mpk_{ZK}||M_0$ where $mpk_{TC}, mpk_{\Delta}, mpk_{ZK}$ are respectively TC's, $\Delta_{Blind-UKG}^{\Pi}$'s and an IA-NMZPoK protocol (see below)'s master public key. Ψ works as follows. For intuition it is also

presented in a figure where the IA-NMZPoK protocol’s arrow points from the prover to its verifier.

Equijoin Protocol Ψ : General Construction

- (1) P_1 computes Π ’s master public/secret-key $(mpk, msk) \leftarrow ESetup(k)$, for each $(u_i, x_i) \in X_1 (i = 1, \dots, N_1)$ computes ciphertext $\xi_i \leftarrow E(mpk, u_i, x_i \| M_0; r_i)$ where r_i is the independent randomness in each encryption, then computes $(cmt, dmt) \leftarrow Cmt(mpk_{TC}, P_2, \xi_1 \| \dots \| \xi_{N_1})$ and sends $mpk \| cmt$ to P_2 .
- (2) P_1 and P_2 run the protocol $\Delta_{Blind-UKG}^\Pi$ where P_1 (as the key-generator) inputs (mpk, msk) and P_2 (as the key-receiver) inputs v_1, \dots, v_N to $\Delta_{Blind-UKG}^\Pi$. On $\Delta_{Blind-UKG}^\Pi$ ’s completion, P_1 obtains N and P_2 obtains $usk(v_1), \dots, usk(v_{N_2})$ as the output.
- (3) P_1 sends $\xi_1 \| \dots \| \xi_{N_1} \| dmt$ to P_2 .
- (4) P_2 verifies $Vf(mpk_{TC}, P_2, \xi_1 \| \dots \| \xi_{N_1}, cmt, dmt) = 1$.
- (5) P_1 runs the protocol IA-NMZPoK $((u_i, x_i, r_i) : \xi_i = E(mpk, u_i x_i \| M_0; r_i), i = 1, \dots, N_1)$ as a prover with P_2 as a verifier. On this IA-NMZPoK’s completion, P_2 tries to decrypt each ξ_i by $usk(v_j)$ ’s it obtained in step 2 and everytime the output has suffix M_0 , i.e., $D(mpk, usk(v_j), \xi_i) = x_i \| M_0$, it generates an entry (v_j, x_i) . All such entries constitute a temporary table X_0 , then P_2 performs a local join $Y_0 \leftarrow Join(w : X_0, X_2)$.
- (6) P_1 outputs N and P_2 outputs Y_0 .

Note that $X_0 = \{(v_j, x_i) \in X_1 : \text{there exists } \xi_i \text{ s.t. } D(mpk, usk(v_j), \xi_i) = x_i \| M_0\}$ and $D(mpk, usk(v_j), \xi_i) = x_i \| M_0$ implies $u_i = v_j$ with negligible exception, so $Y_0 = Join(w : X_1, X_2)$, i.e., Ψ ’s output is correct with only negligible exception probability.

Ψ is actually a $\Delta_{Blind-UKG}^\Pi$ -hybrid protocol and we require $\Delta_{Blind-UKG}^\Pi \rightarrow^{GUC} F_{Blind-UKG}^\Pi$ (definition 2.1). However, merely requiring $\Delta_{Blind-UKG}^\Pi \rightarrow^{GUC} F_{Blind-UKG}^\Pi$ cannot guarantee Ψ ’s GUC-security but only “half GUC-security” instead (i.e., the real adversary A corrupting P_1 can be completely simulated by an ideal adversary S but this is not true when A corrupts P_2 . Only data-privacy can be proved in the latter case). In order to make the real adversary completely simulatable in ideal-world, some additional property is required for $\Delta_{Blind-UKG}^\Pi$. This leads to definition 3.1 and it is not hard to verify that our concrete construction of $\Delta_{Blind-UKG}^\Pi$ in next section really satisfies it.

Definition 3.1 (IBE’s User Private-keys Blind Generation Protocol with Extractor.) Given IBE scheme $\Pi = (ESetup, UKG, E, D)$ and $\Delta_{Blind-UKG}^\Pi \rightarrow^{GUC} F_{Blind-UKG}^\Pi$, let P_1, P_2 be $\Delta_{Blind-UKG}^\Pi$ ’s parties where P_2 provides user-id a and obtains $usk(a)$, P_1 owns msk and (blindly) generates $usk(a)$ for P_2 . This $\Delta_{Blind-UKG}^\Pi$ is defined as *extractable*, if there exists P.P.T. algorithm $Setup_\Delta, UKG_\Delta, Ext_\Delta = (Ext_1, Ext_2)$ and a negligible function $\delta(k)$, called the *error function*, such that

- (1) $Setup_\Delta(k)$ generates the master public/secret key-pair (mpk_Δ, msk_Δ) .
- (2) $UKG_\Delta(msk_\Delta, id)$ outputs a trapdoor $usk_\Delta(P_2)$ when $id = P_2$ (key-receiver’s

identity) and outputs nothing otherwise.

(3) for any user-id a , honest P_1 and any P.P.T. algorithm A , it is true that (via notations in subsection 2.3) $Ext_1(usk(P_2))$ outputs (σ, τ) such that $Pr[Ext_2(mpk||\tau; A(a))_{[\sigma]=a}] > Pr[A_a(mpk; P_1(mpk, msk))_{[\sigma]=UKG(msk, a)}] - \delta(k)$

where (mpk, msk) is Π 's master public/secret-key owned by P_1 (mpk is published).

We stress that all extractors in definition 2.3 and definition 3.1 are non-rewinding.

Combining all the instantiations of subprotocols in this general construction (details presented in Appendix C and D of this paper's full version at eprint.iacr.org/2009/204), it's easy to see that we can get a $O(1)$ and $O(N_1 + N_2)$ message-complexity solution. The exact computation efficiency analysis can be only done for specific instantiations.

Theorem 3.1. *Suppose that IBE scheme $\Pi=(ESetup, UKG, E, D)$ is both id-selective ANO_CPA anonymous and id-selective IND_CPA data-private, $\Delta_{Blind-UKG}^\Pi \rightarrow^{GUC} F_{Blind-UKG}^\Pi$ with extractor $Ext_\Pi=(Ext_{\Pi,1}, Ext_{\Pi,2})$ and error function δ as in def.3.1, IA-NMZPoK $((u_i, x_i, r_i) : \xi_i = E(mpk, u_i, x_i||M_0; r_i), i = 1, \dots, N_1)$ is an IA-NMZPoK protocol, $TC = (D, TSetup, UKG, Cmt, Vf, FakeCmt, FakeDmt)$ is an IBTC scheme, then $\Psi \rightarrow^{GUC} F_{INT}$ assuming static corruptions.*

Proof. We prove the GUC-security in two cases that the real-world adversary A corrupts P_1 or P_2 respectively. Below P_1^* and P_2^* stand for P_1 and P_2 's respective counterparts in ideal-world.

All parties are assumed to be initialized with a copy of the common reference string $ACRS$, i.e., the concatenation of TC 's master public-key mpk_{TC} , $\Delta_{Blind-UKG}^\Pi$'s mpk_Δ , the IA-NMZPoK protocol's mpk_{ZK} and M_0 , generated by the pre-setup G_{ACRS} . For this $ACRS$, its $msk = msk_{TC}||msk_\Delta||msk_{ZK}$ and $UKG(msk, id)$ responses with $usk(id) = usk_{TC}(id)||usk_\Delta(id)||usk_{ZK}(id)$ where $usk_{TC}(id)$, $usk_\Delta(id)$ and $usk_{ZK}(id)$ are respectively TC 's, $\Delta_{Blind-UKG}^\Pi$'s and the IA-NMZPoK protocol's user private-keys corresponding to $id \in \{P_1, P_2\}$.

(1) A corrupts P_1 : for simplicity we first make the proof in $F_{Blind-UKG}^\Pi$ -hybrid model and then complete the proof by generalized universal composition theorem. Let $X_1 = \{(u_1^*, x_1^*), \dots, (u_{N_1}^*, x_{N_1}^*)\}$ be A 's (i.e., P_1 's) own table, $X_2 = \{(v_1^*, y_1^*), \dots, (v_{N_2}^*, y_{N_2}^*)\}$ be P_2^* 's own table. We need to construct an ideal adversary S_1 who corrupts P_1^* , runs A as a black-box and simulates the real-world honest party P_2 to interact with A :

On receiving the message (sid, "input", N_2) from F_{INT} , S_1 gets $usk(P_1)$ by querying the shared functionality G_{ACRS} with ("retrieve", sid, P_1) where $usk(P_1) = usk_{TC}(P_1)||usk_\Delta(P_1)||usk_{ZK}(P_1)$, computes $(\sigma, s, \tau) \leftarrow$ IA-NMZPoK::

$Ext_1(usk_{ZK}(P_1))$ (to avoid ambiguity, we use $\Gamma :: f$ to represent a protocol Γ 's algorithm f), generates N_2 entries $(v_1, y_1), \dots, (v_{N_2}, y_{N_2})$ at random and then starts A ;

After A sends the first message $(mpk\|cmt)$, S_1 interacts with A as an honest key-receiver in model of $F_{Blind-UKG}^{\Pi}$ and obtains $usk(v_1), \dots, usk(v_{N_2})$;

S_1 intercepts the message $\xi_1\|\dots\|\xi_{N_1}\|dmt$ sent from A , verifies whether $Vf(mpk_{TC}, P_2, \xi_1\|\dots\|\xi_{N_1}, cmt, dmt) = 1$ and then participates in protocol IA-NMZPoK($(u_i^*, x_i^*, r_i) : \xi_i = E(mpk, u_i^*, x_i^* \| M_0; r_i), i = 1, \dots, N_1$ as a verifier calling the knowledge extractor IA-NMZPoK:: $Ext_2(\tau)$ to extract the witness $(u_i^*, x_i^*, r_i), i = 1, \dots, N_1$ (in fact only u_i^* 's and x_i^* 's are needed in this proof);

S_1 sends the message (sid, "input", $\{(u_1^*, x_1^*), \dots, (u_{N_1}^*, x_{N_1}^*)\}$) to F_{Join} , then outputs whatever A outputs to the environment.

Let $tr(A, S_1)$ denote the transcripts due to the interaction between S_1 and A , $tr^\psi(A, P_2(X_2))$ denote the transcripts due to the interaction between A and $P_2(X_2)$ in the real-world protocol $\Psi(P_2(X_2))$ means the real-world party possessing the same private set X_2 as P_2^*). From A 's perspective, the difference between $tr(A, S_1)$ and $tr^\psi(A, P_2(X_2))$ is that the former provides $F_{Blind-UKG}^{\Pi}$ with $\{v_1, \dots, v_{N_2}\}$ as the input, the latter provides $F_{Blind-UKG}^{\Pi}$ with $\{u_1^*, \dots, v_{N_2}^*\}$, but according to $F_{Blind-UKG}^{\Pi}$'s specification A knows nothing about what data-entries are provided to $F_{Blind-UKG}^{\Pi}$ by the other party except the number N_2 , as a result, $tr(A, S_1) \approx tr^\psi(A, P_2(X_2))$ (perfectly indistinguishable) from A 's perspective. In particular, the distribution of A 's output due to interactions with S_1 is the same as that (in real-world protocol Ψ) due to interactions with $P_2(X_2)$. Let η be IA-NMZPoK protocol's error function, $Adv_{TC}^{binding}$ be attacker's advantage against TC's binding property, all are negligible functions in k . It's not hard to show (by contradiction) that the probability with which S_1 correctly extracts all A 's data-entries $(u_1^*, x_1^*), \dots, (u_{N_1}^*, x_{N_1}^*)$ is greater than the probability $Pr[P_2(mpk\|\xi_1\|\dots\|\xi_{N_1}; A) = 1] - N_1(\eta + Adv_{TC}^{binding}) \geq Pr[P_2 \text{ outputs } Join(w : X_1, X_2)] - N_1(\eta + Adv_{TC}^{binding})$, therefore, the difference between the probability with which $P_2^*(X_2)$ outputs $Join(w : X_1, X_2)$ under the ideal-world adversary S_1 's attacks and the probability with which $P_2(X_2)$ outputs $Join(w : X_1, X_2)$ under the real-world adversary A 's attacks against Ψ is upper-bounded by $N_1(\eta + Adv_{TC}^{binding})$, also a negligible function in k . Combining all the above facts, for any P.P.T. environment Z we have $output_Z(\Psi, A) \approx^{PPT} output_Z(F_{Join}, S_1)$, i.e., $\Psi \rightarrow^{GUC} F_{Join}$ in $F_{Blind-UKG}^{\Pi}$ -hybrid model.

Now replace the ideal functionality $F_{Blind-UKG}^{\Pi}$ with $\Delta_{Blind-UKG}^{\Pi}$ in Ψ . By what is just proved, the assumption $\Delta_{Blind-UKG}^{\Pi} \rightarrow^{GUC} F_{Blind-UKG}^{\Pi}$ and the GUC-theorem, we still have the GUC-emulation consequence. In addition, it's not hard to estimate S_1 's time complexity $T_{S_1} = T_A + O(N_2 + N_1 T_e)$ where T_A and T_e are A 's and the knowledge extractor's computation time.

(2) A corrupts P_2 : Denote A 's (i.e., P_2 's) own table as $X_2 = \{(v_1^*, y_1^*), \dots, (v_{N_2}^*, y_{N_2}^*)\}$, P_1 's own table as $X_1 = \{(u_1^*, x_1^*), \dots, (u_{N_1}^*, x_{N_1}^*)\}$, we need to construct an ideal adversary S_2 . S_2 corrupts P_2^* , gets $usk(P_2)$ by querying the pre-setup G_{ACRS} with ("retrieve", sid, P_2) where $usk(P_2) = usk_{TC}(P_2)\|usk_{\Delta}(P_2)\|usk_{ZK}(P_2)$, generates $(\sigma, s) \leftarrow \text{IA-NMZPoK}::Sim_1(usk_{ZK}(P_2))$, runs A as a black-box and simulates the real-world honest party P_1 to interact with A :

On receiving message (sid, “input”, N_1) from F_{Join} , S_2 generates $(u_1, x_1), \dots, (u_{N_1}, x_{N_1})$ at random, computes $(mpk, msk) \leftarrow Setup(k)$ and $\xi_i \leftarrow E(mpk, u_i, x_i \| M_0; r_i)$ for each (u_i, x_i) where r_i is the independent randomness in each encryption, computes $(cmt^0, \lambda) \leftarrow FakeCmt(mpk_{TC}, P_2, usk_{TC}(P_2))$, starts A and sends the message $mpk \| cmt^0$ to A ;

S_2 interacts with A as the user private-key generator in $\Delta_{Blind-UKG}^{\Pi}$ and calls the extractor $\Delta_{Blind-UKG}^{\Pi} :: Ext_{\Delta}(usk_{\Delta}(P_2))$ to extract v_1^*, \dots, v_N^* (N is the number of distinct v_i^* 's), generates $\{y_1, \dots, y_N\}$ at random, sends the message (sid, “input”, P_2^* , $\{(v_1^*, y_1), \dots, (v_N^*, y_N)\}$) to F_{Join} ;

S_2 sends the message (sid, “join”, P_2^*) to F_{Join} and gets the response $\{(u_{j_1}^*, x_{j_1}^*, y_{j_1}), \dots, (u_{j_t}^*, x_{j_t}^*, y_{j_t})\}$ (i.e., the equijoin of X_1 and $\{(v_1^*, y_1), \dots, (v_N^*, y_N)\}$, in particular this result implies that there are t - v_j 's such that $u_{j_1}^* = v_{j_1}, \dots, u_{j_t}^* = v_{j_t}$). To simplify the notation, denote this response table as $\{(u_1^*, x_1^*, y_1), \dots, (u_t^*, x_t^*, y_t)\}$.

S_2 computes $\xi_i^* \leftarrow E(mpk, u_i^*, x_i^* \| M_0; r_i^*)$ where r_i^* 's are independent randomness for $i = 1, \dots, t$, replaces arbitrary t ξ_i 's with ξ_i^* 's and keeps other $N_1 - t$ ξ_i 's unchanged, making a new sequence denoted as $\xi_1' \| \dots \| \xi_{N_1}'$, computes $dmt^0 \leftarrow FakeDmt(mpk_{TC}, \xi_1' \| \dots \| \xi_{N_1}', \lambda, cmt^0)$. S_2 sends the message $\xi_1' \| \dots \| \xi_{N_1}' \| dmt^0$ to A , interacts with A by calling $IA-NMZPoK :: Sim_2(\xi_1' \| \dots \| \xi_{N_1}', s)$ where $\xi_i' = E(mpk, u_i^0, x_i^0 \| M_0; r_i')$, $i = 1, \dots, N_1$, $u_i^0 = u_i^* = v_i^*$ and $x_i^0 = x_i^*$ for $i = 1, \dots, t$ and $u_i^0 = u_i$, $x_i^0 = x_i$ for other i 's.

Finally S_2 outputs whatever A outputs to the environment.

Let $tr(S_2, A)$ denote the transcripts due to the interaction between A and S_2 , $tr^{\Psi}(P_1(X_1), A)$ denote the transcripts due to the interaction between A and the real-world party $P_1(X_1)$ (possessing the same set $X_1 = \{(u_1^*, x_1^*), \dots, (u_{N_1}^*, x_{N_1}^*)\}$ as the ideal-world party P_1^*). From A 's perspective, the differences between these two transcripts are: a) cmt in these two transcripts are respectively cmt^0 output by $FakeCmt$ and cmt output by $Cmt(mpk_{TC}, P_2, E(mpk, u_1^*, x_1^* \| M_0; r_1) \| \dots \| E(mpk, u_{N_1}^*, x_{N_1}^* \| M_0; r_{N_1}))$; b) dmt in these two transcripts are dmt^0 output by $FakeDmt$ and dmt output by $Cmt(mpk_{TC}, P_2, E(mpk, u_1^*, x_1^* \| M_0; r_1) \| \dots \| E(mpk, u_{N_1}^*, x_{N_1}^* \| M_0; r_{N_1}))$ respectively; c) Among the ciphertext sequence $\xi_1 \| \dots \| \xi_{N_1}$ in these two transcripts, there are t ciphertexts ξ_i having the same identity public-key u_i^* and the same plaintext $x_i^* \| M_0$ but the remaining $N_1 - t$ ciphertexts having different identity public-keys; d) there are t IA-NMZPoK-witness' with the same u_i^0 and x_i^0 .

Because of TC's equivocation property, (cmt, dmt) 's are P.P.T.-indistinguishable in both cases; because of IBE scheme's selective ANO-CPA anonymity and IND-CPA data-privacy, $\xi_1 \| \dots \| \xi_{N_1} \| dmt$ in both cases are P.P.T.-indistinguishable (otherwise suppose they are P.P.T.-distinguishable with the difference $\delta \geq 1/poly(k)$, it's not hard to construct either a selective ANO-CPA or IND-CPA attacker against Π with an advantage at least δ/N_1 , contradicting with Π 's either selective ANO-CPA anonymity or data-privacy). Now denote the ciphertext sequence $\xi_1 \| \dots \| \xi_{N_1}$ in two cases as $\xi_1^{(1)} \| \dots \| \xi_{N_1}^{(1)}$ and $\xi_1^{(2)} \| \dots \| \xi_{N_1}^{(2)}$ respectively, denote the transcripts in session of IA-NMZPoK as $IA-NMZPoK^{(1)} (= tr_{S_2, A}(mpk \| M_0 \| \xi_1^{(1)} \| \dots \| \xi_{N_1}^{(1)}))$ and $IA-NMZPoK^{(2)}$

($=tr_{P_{1,A}}(mpk \| M_0 \| \xi_1^{(2)} \| \dots \| \xi_{N_1}^{(2)})$) respectively, by the above analysis we have $\xi_1^{(1)} \| \dots \| \xi_{N_1}^{(1)} \approx^{PPT} \xi_1^{(2)} \| \dots \| \xi_{N_1}^{(2)}$; furthermore, by IA-NMZPoK’s zero-knowledge property we have

$$\text{IA-NMZPoK}^{(2)} \approx^{PPT} \text{IA-NMZPoK}::\text{Sim}_2(\xi_1^{(2)} \| \dots \| \xi_{N_1}^{(2)}, s)$$

and by S_2 ’s construction we also have

$$\text{IA-NMZPoK}^{(1)} = \text{IA-NMZPoK}::\text{Sim}_2(\xi_1^{(1)} \| \dots \| \xi_{N_1}^{(1)}, s)$$

so $\text{IA-NMZPoK}^{(1)} \approx^{PPT} \text{IA-NMZPoK}^{(2)}$.

As a result, the transcripts received by A in both cases are P.P.T.-indistinguishable.

Let δ be $\Delta_{Blind-UKG}^{\Pi}$ ’s extractor’s error function (negligible in k), then the probability with which S_2 correctly extracts A ’s one data-item y_i^* is at least $Pr[A(mpk; P_1(mpk, msk)) = UKG(msk, y_i^*)] - \delta$, so the probability with which S_2 correctly extracts A ’s all values v_1^*, \dots, v_N^* is at least $Pr[A(mpk; P_1(mpk, msk)) = UKG(msk, v_i^*) : i = 1, \dots, N] - N_2\delta \geq Pr[P_2 \text{ outputs } Join(w : X_1, X_2)] - N_\delta$. As a result, S_2 ’s output is P.P.T.- indistinguishable from A ’s output in Ψ with respect to the GUC-environment Z with an error upper-bounded by $N_1(k) Adv_{\Pi}^{ANO-CPA}(k) + N_2\delta(N \leq N_2)$, which is negligible in k . Note that in both cases the other party $P_1^*(X_1)$ and $P_1(X_1)$ always output the same N , so we have the consequence that $output_Z(\psi, A) \approx^{PPT} output_Z(F_{Join}, S_2)$ and it’s easy to estimate S_2 ’s time-complexity $T_{S_2} = T_A + O(N_1 + N_2 T_{ext})$ where T_A and T_{ext} are A ’s and the extractor’s computation-time.

By all the facts, we have $\Psi \xrightarrow{GUC} F_{Join}$.

4 Summary and Extensions

We present a general solution to 2-party join operation in distributed relational database via the secure and anonymous IBE scheme which is provably GUC-secure in standard model with reasonable efficiency. Although there can be other approaches to the same solution, our approach has the good potential to deal with more complicated cases within a unified framework. In fact what we deal with in this paper is just the most simple (also most frequently-used) case: *equijoin*. A general join-operator (called theta-join) involves some condition on attributes from each of its argument tables, i.e., $Join(\theta(\mathbf{a}, \mathbf{b}) : X_1, X_2)$ where θ is a predicate, \mathbf{a}, \mathbf{b} are vectors of attributes of X_1 and X_2 respectively, e.g., $\mathbf{a} = (a^{(1)}, a^{(2)})$, $\mathbf{b} = (b^{(1)}, b^{(2)})$. For equijoin, $\theta(\mathbf{a}, \mathbf{b}) \equiv a^{(1)} = b^{(1)} \wedge a^{(2)} = b^{(2)}$ but in general $\theta(\mathbf{a}, \mathbf{b})$ can be any predicate, e.g., $a^{(1)} < b^{(1)}$, $a^{(1)} + a^{(2)} \geq b^{(1)} + b^{(2)}$, $a^{(1)} < b^{(1)} \wedge a^{(2)} = b^{(2)}$, etc. In future work we will apply the recently proposed ABE schemes(e.g.,[11,17], which are powerful generalizations of IBE) to develop solutions to these secure SQL-operations within the same general protocol framework as in section 3, which has obvious advantages in practice.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Blake, I., Kolenilkov, V.: Conditional Encrypted Mapping and Comparing Encrypted Numbers. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 206–220. Springer, Heidelberg (2006)
3. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-based Encryption without Random Oracles. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
4. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd Annual Symposium on foundations of computer Science, pp. 136–145. IEEE Computer Society, New York (2001) (Updated in 2005), <http://eprint.iacr.org/2000/067>
5. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Security with Global-Setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
6. Dodis, Y., Shoup, V., Walfish, S.: Efficient Constructions of Composable Commitments and Zero-Knowledge Proofs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 515–535. Springer, Heidelberg (2008)
7. Garacia-Molina, H., Ullman, J., Widom, J.: Database System Implementation. Prentice-Hall Inc., Englewood Cliffs (2004)
8. Garay, J., MacKenzie, P., Yang, K.: Strengthening Zero-Knowledge Protocols Using Signatures. *Journal of Cryptology* 19(2), 169–209 (2006)
9. Hacigumus, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over Encrypted Data in the Database-Service-Provider Model. In: ACM SIGMOD 2006, pp. 96–107 (2006)
10. Hazay, C., Lindell, Y.: Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
11. Katz, J., Sahai, A., Waters, B.: Predicate Encrypted Supporting Disjunctions, Polynomial Equations and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
12. Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. *Journal of Cryptology* 15(3), 177–206 (2002)
13. Kissner, L., Song, D.: Privacy-Preserving Set Operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
14. MacKenzie, P., Yang, K.: On Simulation-Sound Trapdoor Commitments. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 382–400. Springer, Heidelberg (2004)
15. Ozsu, T., Valduriez, P.: Principles of Distributed Database Systems, 3rd edn. Prentice-Hall Inc., NJ (2004)
16. Shi, E., Bethencourt, J., Chen, H., Song, D., Perrig, A.: Multi-dimensional Range Queries over Encrypted Data. In: IEEE Symposium on Security and Privacy (2007)
17. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient and Provably Secure Realization. *eArchive* 2008/304 (2008)

TSM-Trust: A Time-Cognition Based Computational Model for Trust Dynamics

Guangquan Xu*, Zhiyong Feng, Xiaohong Li, Hutong Wu, Yongxin Yu, Shizhan Chen, and Guozheng Rao

School of Computer Science and Technology,
Tianjin University, China

{xuguangquan, zfyfeng, xiaohongli, wht, yyx, shizhan, rgz}@tju.edu.cn

Abstract. This paper proposes a hierarchical network model for trust evaluation after introducing time cognition, which mainly considers trust dynamics. In this model, the Temporal Sequential Marker (TSM) is tagged on each item in an implicit or explicit manner and all items are divided into several layers according to their TSMs information. Furthermore, three different kinds of forgetting effects are investigated and quantified for the computing of TSM-Trust. These effects are: distance effect, boundary effect and hierarchical effect. Next, according to the Ebbinghaus curve of forgetting, cosine function is used to model the forgetting process of Experience Information (EI) approximately, the D-S theory is exploited to build up a computational dynamic trust (TSM-Trust) model based on our proposed hierarchical network model. Finally, our future work is pointed out after analyzing the limitations of this paper.

Keywords: TSM, trust evaluation, Dempster-Shafer theory of belief functions, Ebbinghaus curve of forgetting, hierarchical network model.

1 Introduction

It is out of discussion that the importance of trust and reputation in human societies is realized [1]. As a kind of informal social capital, trust has been playing a unique role in maintaining the stability of societies and driving its progress, and especially nowadays its function is even irreplaceable. Trust and reputation has been regarded as one of the most important elements in accelerating transactions and fostering markets in Virtual Organizations (VOs). Furthermore, people have not stopped researching and probing into trust. So far, most researches can be classified into the following: The one that focuses on the concepts of trust, that is "what is trust"; another emphasizes on trust modeling and reputation (i.e. trust management). The last one considers trust decision-making.

Historically, although some contributions on trust dynamics have been contributed, to this date only a few of them have taken into account its reliance

* Please note that the LNCS Editorial assumes that all authors have used the western naming convention, with given names preceding surnames. This determines the structure of the names in the running heads and the author index.

to time cognition and focus on trust dynamics. With reference to what we have just highlighted therefore, we will begin this paper with an extensive review and discussions on related literatures. Our main contributions are as follows:

- Interaction experiences includes direct and indirect information, which are regarded as series of items with time information (time information is represented by Temporal Sequential Marker). As far as trust computing is concerned, experiences used to reason the value of trust are imputed to the related information tagged with TSM.
- After the process of tagging information with TSM is achieved, each item is then classified into different groups to form a hierarchical network. Next we discuss the trust reasoning mechanism for each agent, based on which we construct a conceptual model of trust computation on its attribution of dynamics.
- After that we explain its logical method in weaving the hierarchical network and introduce three different kinds of forgetting effects: distance effect, boundary effect and hierarchical effect.
- Considering its fuzzy semantics, trust is fit to be dealt with by such uncertain mathematics theories as Dempster-Shafer theory of belief functions. Here D-S theory is expanded to be used for computing the values of trust in dichotomy (i.e. trust or distrust) situation.

In section 2, we review the related work on trust dynamics and discuss their advantages and disadvantages respectively. We introduce time cognition and Temporary Sequential Marker (TSM) and then consider applications in various multi-agent systems. Section 3 constructs a conceptual Hierarchical Network Model based on TSM. In Section 4, a computational trust model is proposed based on D-S theory. Such a D-S theory model mainly concerns the fuzzy semantics of trust dynamics. Finally, section 5 holds our conclusions and points out our future work.

2 Related Work

Although a few works have attempted to address the dynamics of trust, less of them focus on the attribution of time cognition, that is to say, there are nearly no works which research trust dynamics from a cognitive view. Next we will introduce some outstanding work about trust dynamics.

Liu et al [2] propose a temporal logical belief for specifying the dynamics of trust for multi-agent systems. In their opinion, trusting someone means having beliefs for a given goal in some fixed environment, and the reason for trust dynamics lies in its continuous changing environments. Therefore temporal logical belief is used to specify this dynamics. It is one of the earlier works in considering temporal features and introducing temporal dimensions to model evolving theories of trust for multi-agent systems through proposing TML (Typed Modal Logic, which extends classical first-order logic with typed variables and multiple belief modal operators.). Furthermore, they examine two main aspects of trust

dynamics: a) How direct experiences involving trust, with their successes or failures, influences the future trust of an agent about similar facts. The authors believe that a cognitive attribution process is needed in order to update trust on the basis of 'interpretation' of the outcome of interactions between them (failure or success); b) How the fact that A trusts B and relies on it in a situation can actually (objectively) influence B's trustworthiness in the Ω situation.

Literature [3] provides a method for modeling the dynamics of trust within a system, which includes a technique for expressing trust changes at a given trust state, and an abstract algorithm for obtaining the new trust state from a given state and a trust change.

Another work is from Chang Jun-Sheng et al [4], their work presents a time-frame based dynamic trust model DyTrust. After incorporating time dimension using time-frame, the authors also introduce four trust parameters in computing trustworthiness of peers. Altogether, these parameters are adjusted in time to reflect the dynamics of trust environment using feedback control mechanism, thus trust evaluation has better adaptability to the dynamics of trust.

Zhang Wei et al [5] point out that the dynamic nature of trust creates the biggest challenge in measuring trust and predicting trustworthiness, therefore they introduce the theory of Fuzzy Cognitive Time Maps (FCTMs) into modeling and evaluating trust relationships and show how relevant is the inter-organizational trust based on trust sources and their credibility. This is a most recent work in considering the time cognition of trust.

Advances in network and microprocessor technology have increased the adoption of computer technology in areas such as consumer shopping, banking, voting, and automotive technology. In the meanwhile, the general public has been aware of the following risk. Trust is playing a crucial role both in the successful introduction of new products and services (including computer technology) and the evolution of intelligent vehicles [6]. However, trust is a dynamic phenomenon in its intrinsic nature, whether in the human society or in the computer-based virtual community. Trust changes with experience, with the modification of the different sources it is based on, with the emotional state of the trustor, with the modification of the environment in which the trustee is supposed to perform, and so on. In other words, since trust is an attitude depending from dynamic phenomena, it is itself a dynamic entity [3]. There are some other studies on the dynamics of trust [7][8].

3 TSM Based Hierarchical Network Model

Information must happen at a given history time, so time is attached to it when coding for each information, either in an implicit or explicit manner. As far as trust computing is concerned, experiences used to reason the value of trust are imputed to the related information tagged with TSM. In this way, we embark on the representation of trust dynamics.

In fact, TSM is a kind of structured social annotation. As a form and tool for network resources or documents, it is widely popular among more and more

researchers. In the meanwhile, as a form of knowledge discovery, sharing and cooperating, it also embodies the spirit of web 2.0, and hence has been paid more and more attention to. TSM is tagged to EI so as to model the dynamics of trust.

All TSMs of items form an organization with hierarchical network. As we know, TSM can be any symbolic system with fixed order, e.g. 1, 2, 3 n; a, b, c . . . z. According to Li Boyue et al [9], in order to improve cognitive efficiency, the items (here is EI) should be divided into groups once the number of them reaches a preset threshold. When there are rich semantics in the items, we can depend on the semantic relationship to divide them. Otherwise what we can rely on is TSM. Furthermore, as items become more, more layers division is needed, and as a consequence there emerges distance effect, boundary effect and hierarchical effect. In Li's views [9], there is no direction effect. Next we will give a conclusion about distance effect, boundary effect and hierarchical effect mainly based on Li's experimental results.

Proposition 1. When two items of EI are from different groups, there will exist distance effect, boundary effect and hierarchical effect.

- When several items are combined into groups without a clear boundary, distance effect will exist between two items with an interval of other two items. That is to say, if some EIs for trust are combined, distance effect will exist between any two EIs with an interval of other two EIs.
- When several items are combined into groups with a clear boundary, the boundary effect will exist between two items in different groups. When two items in different groups are combined to compute the value of trust, boundary effect will exist.
- When hierarchical network is formed, the hierarchical effect will exist between two items of EI in different levels.

Of course, how to construct a hierarchical network of EI based TSM is a puzzle, and in most cases in human society it is formed automatically and implicitly. But in e-society or so called computer-centered system, it must be carried out artificially. Therefore different subjective judgment of each individual will impact on three effects derived by distance and boundary dramatically. In this paper, as a TSM tool, calendar time is used to construct a hierarchical network based on TSM, and trust computed and reasoned from such a hierarchical network is called TSM-Trust based on TSM-HN. How distance and boundary and hierarchical effects will impact on the value of trust will be discussed in section 4.

When several items are assembled as a group without an indefinite boundary, distances effect will exist between two items with an interval of other two items. When there is definite boundary between such groups, boundary effect will exist between two items in different groups. And lastly when a hierarchical network is formed, hierarchical effect will exist between two items in different layers. In fact, these effects are same in nature, i.e. all of them belong to time forgetting effects. Each TSM locates at one position of this network, and also owns the corresponding code to represent hierarchical network. The order of two items

is compared by searching the positions of their representations (TSMs) in the hierarchical network.

Comparing with the semantic hierarchical network model initiated by Collins and Quillian [10], TSM based Hierarchical Network (TSM-HN) has the following characteristics, which are also reasons why we choose TSM-HN as our reasoning foundation.

- TSM-HN's representation has automatic embedded and implicit features. Firstly, any item happens at a given time, therefore time information is tagged upon this item automatically and implicitly. That is to say, there is no need to worry about losing time information of each item, because the time information is stored in system logs automatically. Secondly, when dealing with item with TSM, it is implicit that items are divided into chunks, groups and levels. Of course, as far as trust evaluation is concerned, EI (including direct information and indirect information) is tagged with time information intentionally and artificially, and then just because of its automatics of TSM-HN, each EI is existent together with its TSM all the time.
- The fuzzy characteristic of the representation of TSM-HN. Individual can deal with TSM-HN in a fuzzy way, which results into fuzzy boundary problem of TSM-HN. For a system with continuous sequential markers, there is not a line of definite boundary. "1" and "3" can belong to one group, "3" and "5" can belong to another group, "5" and "7" can belong to another group, but "1" and "5" can not belong to the same group, "1" and "7" can not either. Conversely, "5" and "3" can belong to one group, and "5" and "7" can belong to another group. In a word, human must divide all items into more layers, but no boundary line can be found, such a moving line in fact is a fuzzy line, it is dependent on individual subjective judgment. That is one of the most important reasons for our exercising D-S theory to compute the value of trust.
- The capacity of the representation of TSM-HN is also fuzzy. In order to improve the efficiency of our searching for TSM, we also need make the capacity of each group consistent with the capacity of time memory of human, although computer system has much more capacity of memory and storage. For human, what is the biggest capacity of a group? To answer this question, one point should be mentioned: because of the limitation of the capacity of human's work memory, one group can contain at most seven items. Broadbent et al [11] point out each group can contain no more than three items. However Li Boyue et al [10] believe that one group can contain at most four to five items after repeated experiments, which is in conflict with results from Broadbent et al. Simply put, such a capacity of one group is also fuzzy, it relies on individual age and other individual differences. In this paper, we assume this capacity is seven, which is the limitation of the capacity of work memory and can improve the efficiency of searching TSM information.
- The subjectivity of the representation of TSM-HN. Objective time is linear, TSM-HN is formed artificially, hence is subjective. The above automatics,

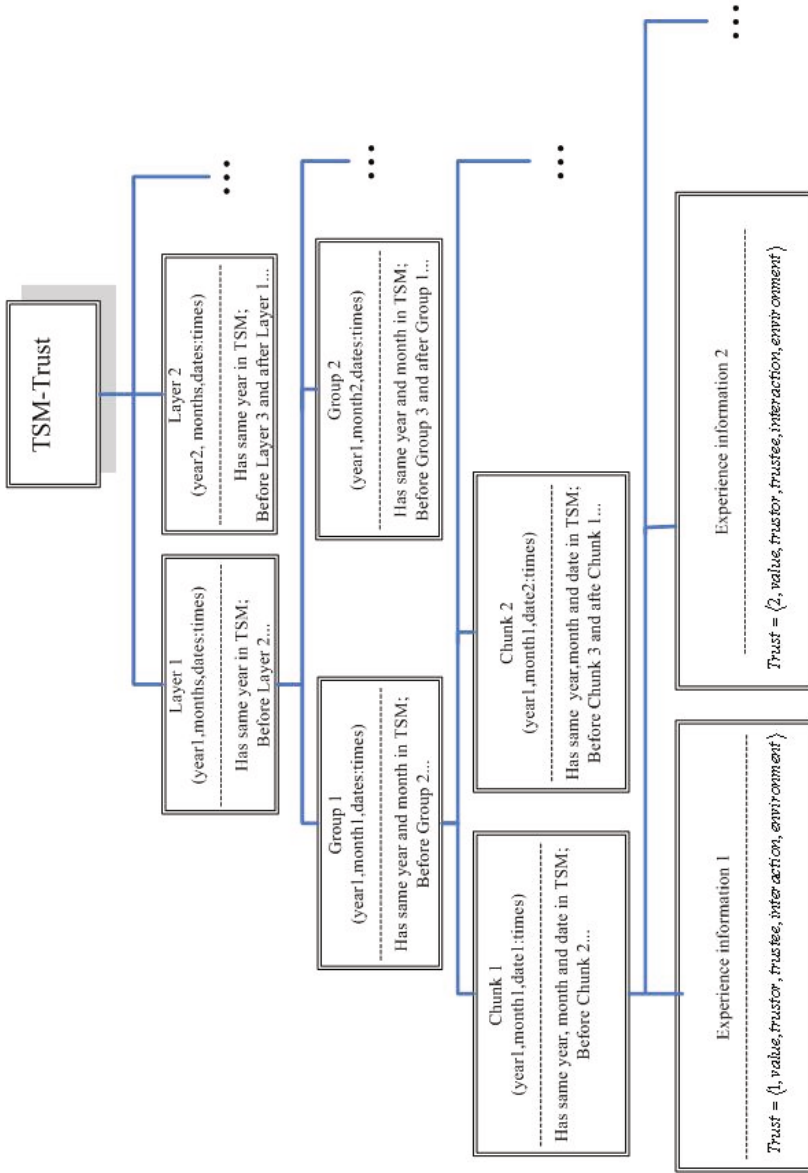


Fig. 1. The trust reasoning model based on TSM-HN

implicit, fuzzy and capacity and so on, are all resulted from the subjectivity. More importantly, in Li's [10] views, even "objective" TSM-HN organization coded by calendar time is limited by this subjectivity.

- The forgetting effects mentioned above exist in TSM-HN. Bigger the interval between items is, more obvious the forgetting effect becomes.

In this paper, trust dynamics involving the time information of interaction is paid more attention to. As for space dimension, much work has pointed out trust can be classified into direct trust and indirect trust, which are derived from direct experience and indirect experience information respectively. Like some work considering time dimension, here short time trust and long time trust are also regarded as the most two kinds of trust.

Our model is two-dimensional metrics of time-space, so although there are many unique aspects including considering both time and space characteristics concurrently, some work is based upon previous achievements from either time dimension or space dimension respectively. As other models mentioned above, trust is computed by EI (including direct information and indirect information, the latter is also called recommending information). This kind of trust is reasoned from space dimension, and it can only represent the attribution of space; as for the time attribution, the following trust reasoning model is built up based on TSM-HN[Fig.1].

4 Computational Model Based on TSM-HN and Its Algorithm

Ebbinghaus is famous for the Ebbinghaus curve of forgetting effect [12]. There is still much evidence which have proved that memory is changing with time in a curve of attenuation trend. Since TSM is used to represent time information of EI, when assessing the trust of given agent, our reasoning model shown in Figure 1 becomes our main foundation. What's more, actually time characteristics is reduced to a problem of space through introducing TSM-HN, then first trust is considered from space dimension aspect. As some research work has pointed out that trust is classified into direct trust and indirect trust, and they are also combined to compute the value of trust. Here the rating of trust is defined as follows:

Definition 1. Space Trust (Space-Trust) is made up of direct trust (DT_{ij}) and indirect trust (IT_{ij}), that is

$$(\text{Space} - \text{Trust})_{ij} = f(DT_{ij}, IT_{ij}) = \lambda \times DT_{ij} + (1 - \lambda) IT_{ij} \quad (1)$$

Where λ is the preference factor, whose value depends on much more complicated factors than expected. As in human society, each agent focuses on direct experience while pays less attention to indirect experience. Therefore although the value of λ can not be confirmed concisely, its value always lies above 0.5. According to trust transfer mechanism and trust clustering mechanism [13], direct trust (DT_{ij}) and indirect trust (IT_{ij}) are defined as follows.

Definition 2. Direct Trust $DT_{ij} = [Bel_{ij}(\{T\}), Pl_{ij}(\{T\})]$ can be computed in that all EIs are looked as parallel information, and then they can be combined by Shafer’s rule of combination.

Based on the Shafer’s rule of combination [14],
 $m_{ij} = m_1(i, j) \oplus m_2(i, j) \oplus \dots \oplus m_n(i, j)$

If evidence information (namely Experience Information, EI) $e_1(i, j), e_2(i, j), \dots, e_n(i, j)$ have basic probability assignments $m_1(i, j), m_2(i, j), \dots, m_n(i, j)$, and the corresponding belief intervals are noted as $DT_{ij}^k = [Bel_k(\{T\}), Pl_k(\{T\})]$, $1 \leq k \leq n$. Then according to trust clustering mechanism [13], direct trust between agent i and agent j is:

$$DT_{ij} = \bigoplus_{k=1}^n DT_{ij}^k = \bigoplus_{k=1}^n [Bel_k(\{T\}), Pl_k(\{T\})], 1 \leq k \leq n \tag{2}$$

Definition 3. Indirect Trust can be gotten approximately by regarding each recommendation router as series case, then regarding all routers as a parallel case, that is to say, indirect trust (IT_{ij}) can be computed as follows:

Assuming that there are μ routers between i and j , and there is no less than one node. For any given router ν , if there are p nodes between i and j , i.e.

$$\nu = \{i \rightarrow r \rightarrow r + 1 \rightarrow \dots \rightarrow r + p - 2 \rightarrow r + p - 1 \rightarrow j\}$$

Then

$$Bel_{ij}^\nu(\{T\}) = Bel_{i,r}(\{T\}) Bel_{r,r+1}(\{T\}) \dots Bel_{r+p-2,r+p-1}(\{T\}) Bel_{r+p-1,j}(\{T\}) \tag{3}$$

$$Pl_{ij}^\nu(\{T\}) = Pl_{i,r}(\{T\}) Pl_{r,r+1}(\{T\}) \dots Pl_{r+p-2,r+p-1}(\{T\}) Pl_{r+p-1,j}(\{T\}) \tag{4}$$

So we can get indirect trust between i and j transferred by router ν :

$$IT_{ij}^\nu = [Bel_{ij}^\nu(\{T\}), Pl_{ij}^\nu(\{T\})] \tag{5}$$

Since all routers between i and j are regarded as parallel relationships, their combination can use trust clustering mechanism [13].

$$\begin{aligned} IT_{ij} &= [Bel_{ij}(\{T\}), Pl_{ij}(\{T\})] \\ &= \bigoplus_{\nu=1}^{\mu} IT_{ij}^\nu = IT_{ij}^1 \oplus IT_{ij}^2 \dots \oplus IT_{ij}^\nu \oplus \dots \oplus IT_{ij}^\mu \\ &= \bigoplus_{\nu=1}^{\mu} [Bel_{ij}^\nu(\{T\}), Pl_{ij}^\nu(\{T\})] \\ &= [Bel_{ij}^1(\{T\}), Pl_{ij}^1(\{T\})] \oplus [Bel_{ij}^2(\{T\}), Pl_{ij}^2(\{T\})] \dots \\ &\quad \oplus [Bel_{ij}^\nu(\{T\}), Pl_{ij}^\nu(\{T\})] \oplus \dots \oplus [Bel_{ij}^\mu(\{T\}), Pl_{ij}^\mu(\{T\})] \end{aligned} \tag{6}$$

Definition 4. TSM based Dynamic Trust (TSM-Trust) is classified into two kinds of trust: non-forgetting trust(trust without forgetting effect), trust with forgetting effect (including trust with distance effect, trust with boundary effect and trust with hierarchical effect). For any two nodes (i, j) , if there is two pieces of EI (e_1, e_2) to support the same proposition, then according to the Ebbinghaus

curve of forgetting, cosine function is exploited to model trust dynamics in time dimension.

$$(TSM - Trust)_{ij} = f \left((Space - Trust)_{ij}, dt \right) = \begin{cases} (Space - Trust)_{ij}, & \text{no forgetting trust} \\ g(dt) \times (Space - Trust)_{ij}, & \text{otherwise} \end{cases} \tag{7}$$

And

$$g(dt) = \cos \left(\frac{\pi(dt)}{2T_{max}} \times \gamma \right) \tag{8}$$

Where $g(dt)$ is called function of forgetting, $dt = |t_\varphi - t_\phi|$ is the time interval between two pieces of current EI(e_1, e_2); T_{max} is the maximal time interval which the trustor can tolerate or memorize trustee (here means EI between nodes i, j); $g(dt)$ is called "the attenuation function of trust", it is modelled by cosine function about the dispersion of time [15]; what's more, in order to normalization, then

$$\begin{aligned} \text{for } \frac{x}{dt} &= \frac{\pi/2}{T_{max}} \\ \text{then } x &= \frac{\pi(dt)}{2T_{max}} \end{aligned} \tag{9}$$

γ is the forgetting factor, its value is assumed as:

$$\gamma = \begin{cases} 1.5, & \text{there is distance effect} \\ 2, & \text{there is boundary effect} \\ 3, & \text{there is hierarchical effect} \\ 0, & \text{there is no forgetting effect} \end{cases} \tag{10}$$

That is to say, when two EIs with an interval of two other chunks in different chunks are combined, distance effect will exist, i.e. $\gamma = 1.5$; when two EIs in different groups are combined, boundary effect will exist, i.e. $\gamma = 2$; when two EIs in different layers are combined, and hierarchical effect will exist, i.e. $\gamma = 3$. Of course, the value of γ is dependent upon trustors' subjective judgment according to their experience knowledge or other factors like trust preference, here we choose a simple assignment for simplicity.

Since the domain of cosine function is limited within $[0, \frac{\pi}{2}]$, that is

$$\begin{aligned} \text{for } 0 &\leq \frac{\pi(dt)}{2T_{max}} \times \gamma \leq \frac{\pi}{2} \\ \text{then } 0 &\leq dt \leq \frac{T_{max}}{\gamma} \end{aligned} \tag{11}$$

and $\cos(\frac{\pi}{2}) = 0$, When $dt \geq \frac{\pi}{2}$, the attenuation function of trust equals zero. Then (Eq. 7) becomes:

$$(TSM - Trust)_{ij} = f \left((Space - Trust)_{ij}, dt \right) = \begin{cases} (Space - Trust)_{ij}, & \text{no forgetting trust} \\ g(dt) \times (Space - Trust)_{ij}, & 0 \leq dt < \frac{T_{max}}{\gamma} \\ 0, & dt \geq \frac{T_{max}}{\gamma} \end{cases} \tag{12}$$

The above work is devoted to the computation of TSM-T, and however it can only be used in situation with two pieces of EI and does not consider the situation with more than two pieces of EI, so modification about should be given out: $dt = |t_{\max} - t_{\min}|$, here t_{\max} is the maximum time (more close to T_{\max}) among all TSMs; t_{\min} (more far from T_{\max}) is the minimum time among all TSMs.

Then TSM-Trust can be computed by the following:

$$(TSM - Trust)_{ij} = f \left((Space - Trust)_{ij}, dt \right) = \begin{cases} (Space - Trust)_{ij}, & \text{no forgetting trust} \\ \cos \left(\frac{\pi(t_{\max} - t_{\min})}{2T_{\max}} \gamma \right) (Space - Trust)_{ij}, & \text{otherwise} \\ 0, & dt \geq \frac{T_{\max}}{\gamma} \end{cases} \quad (13)$$

It should be noted that all the time information is gotten from TSM of EI based on the TSM-HN. Lastly a method for TSM-Trust computing is gotten till now.

5 Concluding Remarks

To investigate the dynamic evolvement of trust, this study bases on the research of time cognition, which formally constructs a hierarchical network model with time lags as TSM. Next, three different kinds of forgetting effects (distance effect, boundary effect and hierarchical effect) are investigated and quantified for the computing of TSM-Trust. Furthermore, Ebbinghaus curve of forgetting is introduced, and then cosine function is exercised to model the attribution of trust dynamics approximately. And then D-S theory is exploited to build up a computational dynamic trust (TSM-Trust) model based on our proposed hierarchical network model.

Some limitations should nevertheless be underlined. First, our hierarchical network is structured according to the TSM system of calendar time, which becomes the only base of classifying different groups, such as chunks, groups and hierarchical levels. Of course, as an attempt of TSM network, it is of importance in investigating the attribution of time cognition for trust dynamics, however more common paradigm should be found out. Second, when dealing with the quantifying the forgetting effect, cosine function is used, whose rationality should be studied more. Finally, the largest memory time T_{\max} is assumed as seven years, and the value of forgetting factor γ is assigned by 0, 1.5, 2 and 3, both of which may be an arbitrary decision and more related study should be committed. In conclusion, trust dynamics is more complex than expected and much more related work should be carried out till people completely master its natural properties so as to foster and manage trust effectively within whether human society or agent society.

It must be pointed out that our example and experiments are discarded here for the limitation of pages, which will be published in a journal paper soon.

Acknowledgments. The authors would like to thank the editors and anonymous referees for their suggestions and the remarkable improvements they brought to this paper. This Paper has been supported by the National Foundation of China under Grant No.90718023, the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z130, and the key technologies R & D program of Tianjin under Grant No. 08ZCKFGX00700.

References

1. Sabater, J., Sierra, C.: Review on Computational Trust and Reputation Models. In: Artificial Intelligence Review, vol. 24, pp. 33–60. Springer, Barcelona (2005)
2. Liu, C., Ozols, M.A., Orgun, M.: A temporalised belief logic for specifying the dynamics of trust for multi-agent systems. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, pp. 142–156. Springer, Heidelberg (2004)
3. Falcone, R., Castelfranchi, C.: Trust Dynamics: How Trust is influenced by direct experiences and by Trust itself. In: AAMAS 2004, New York, USA, July 19-23 (2004)
4. Chang, J.S., Wang, H.M., Yin, G.: DyTrust: A Time-Frame Based Dynamic Trust Model for P2P Systems. Chinese Journal of Computers 29(8) (August 2006)
5. Zhang, W., Liu, L., Zhu, Y.C.: Using fuzzy cognitive time maps for modeling and evaluating trust dynamics in the virtual enterprises. Expert Systems with Applications 35, 1583–1592 (2008)
6. Hoffman, L.J., Lawson, J.K., Blum, J.: Trust beyond security: an expanded trust model. Communications of the ACM 49(7), 94–101 (2006)
7. Jonker, C., Treur, J.: Formal analysis of models for the dynamics of trust based on experiences. In: AA 1999 Workshop on Deception, Fraud and Trust in Agent Societies, Seattle, USA, May 1, pp. 81–94 (1999)
8. Falcone, R., Castelfranchi, C.: The socio-cognitive dynamics of trust: does trust create trust? In: Falcone, R., Singh, M., Tan, Y.-H. (eds.) AA-WS 2000. LNCS (LNAI), vol. 2246, pp. 55–72. Springer, Heidelberg (2001)
9. Li, B.Y., Huang, X.T.: Research on the representation of time memory: succession & innovation. Xinhua Press, Beijing (2006) (in Chinese)
10. Collins, A.M., Quillian, M.R.: Retrieved time from semantic memory. Journal of Verbal Learning and Verbal Behavior 8, 240–247 (1969)
11. Broadbent, D.E.: The magical number seven after fifteen years. In: Kennney, R.A., Wikes, A. (eds.) Studies in long-term memory. Wiley, NY (1975)
12. Matthew, H.E.: Forgetting and remembering in psychology: Commentary on Paul Connerton's Seven Types of Forgetting (2008), <http://mss.sagepub.com/cgi/content/abstract/1/3/273>
13. Xu, G., Feng, Z., Wu, H., Zhao, D.: Swift Trust in Virtual Temporary System: A Model Based on Dempster-Shafer Theory of Belief Functions. International Journal of Electronic Commerce (IJEC) 12(1), 93–127 (Fall 2007)
14. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
15. Academic Learning Center, Central Piedmont Community College, 103 Garinger Building, Charlotte, NC 28235 704-330-6474 (2008), http://www.cpcc.cc.nc.us/academic_learning/images/sskills/Ebbinghaus.doc

Bring Efficient Connotation Expressible Policies to Trust Management*

Yan Zhang^{1,2}, Zhengde Zhai^{1,2}, and Dengguo Feng^{1,2}

¹ State Key Laboratory of Information Security,
Institute of Software Chinese Academy of Sciences

² National Engineering Research Center of Information Security

Abstract. Trust Management(TM) aims to provide effective access control in open systems. It enables the resource owners to reason and determine the access permissions on the basis of a collection of distributed authorization knowledge about the requester. However, to be efficient, most current TM approaches are based on DATALOG which can't directly express the connotation of TM authorization policies. Thus these policies are hard to be understood and maintained by human beings. In this paper, we propose a new approach called OT based on the ontology language OWL 2 EL. OT supports the connotation expressible policies and remains efficient since its procedure of compliance checking is provable to be tractable.

1 Introduction

The access control in the open distributed system remains to be a challenging problem, and Trust Management(TM) [5] is one of the possible solutions. TM approaches support the distributed authorization and enable the local administrators to make decisions both on local policies and external authorization statements, which are generally in format of signed certifications.

Multitudes of TM approaches with different design features have been studied, such as PolicyMaker [5], KeyNote [4], and RT [13] [14] etc. Existing TM approaches are often composed of a language for describing policies/credentials that state what attribute, role, clearance etc.(referred to as authorization terms from now on) a principal has when he/she has another issued authorization terms, and a compliance checking procedure for deciding whether an access request complies with the submitted credentials and local policies(we use p/c to denote policy/credential from now on).

Every authorization item in a p/c consists of a subject term and several parameter constraints. For example, if an authorization item's subject term is "graduated Student", it may have constraints over three parameters: major, attending university, and GPA. Under human language, these parameter constraints can be free expressed and as complicated as needed. For example, the

* this paper is supported by grants from 863 High-tech Research and Development Program of China (2007AA1204040 and 2007AA1204050).

administrators may define such credentials with “graduated Student” authorization item in the real world:

Example 1.1. The Central Academy of Sciences(CAS) says Alice is “a graduated Student, studying in information security, attending the Institute A, gotten a cumulative GPA of 3.8”

Example 1.2. The Ministry of Education (MOE) says “a CAS graduated student, who is studying in any one of the following fields: information security, software engineering, network,..., and attends any one of the following CAS Institutes: Institute A, Institute B ,..., and has a GPA higher than 3.6” can get an ‘application for becoming an exchange student to VirtualCountry with 5,000 dollars scholarship per year’

Example 1.3. MOE says “a CAS graduated student, who is studying in some fields about computer science, and attends some CAS institute which has cooperated projects with VirtualCountry’s Institutes, and has a GPA which is higher than average” can get an “application for becoming an exchange student to VirtualCountry with 5,000 dollars scholarship per year”

The first case shows a credential of Alice issued by CAS. In this credential, every parameter is constricted to a specific value. In the second credential, the value of every parameter is constricted to be in a predetermined static set. If we call this way in case 1.2 as a denotation constraint description way, then in the last credential, administrator uses a connotation description way to determine a set for every parameter which describes the common property of elements in the set.

Compared with the denotation description way, the connotation constraint description way of authorization items has several advantages in the real world because:

(1) It can reflect the real authorization intention of local administrators accurately. According to some potential security consideration, administrators often can decide which type of parameter values may premise safety and qualification. Writing the “type” instead of plenty of distinct values into the authorization items will make p/c more comprehensible, more concise, and easier to be examined.

(2) It helps the local administrators to create p/c more quickly. It is a time-consuming task to predetermine the parameters’ extension values of authorization items before creating a p/c. What’s worse, when external authorization items are to be included in the new p/c, local administrators have to ask for external entities’ help to figure out the external parameter values pertaining to their thoughts. That premises plenty of time spent for many rounds of communication and negotiation. So, if administrators can write some words standing for the connotation of parameter constraints into p/c directly, the time of predetermining the extension values of those parameter will be greatly reduced.

(3) It makes the created p/c more stable. Under the extension description way, after the extension value sets are determined and the p/c is created, the p/c can’t adapt to any change of the value sets. Actually the sets’ changes may be

frequent in the real world, e.g. the average of GPA in a university will be different when new semester arrives, the number of Institutes which have co-projects with VirtualCountry may increase or decrease daily by daily. Therefore, if there is a method for administrator to express the connotation of parameter constraints, the p/c will maintain stable no matter how frequently the extension value sets are changed.

Motivation. In order to support the connotation description way in TM, a schema to define, publish, share and maintain some machine-readable lexical conceptual structure is needed (so appropriate concept words can be picked to describe the connotation of authorization items in p/c), while a mechanism to match local policies with external credentials not syntactically but semantically (the semantic relation between "computer science" in policy and "information security, network..." in credential can be recognized) is necessary too.

To our best knowledge, there are no existing TM approaches which can be both connotation expressible and decidable. The earliest TM approach—Policymaker is able to express any kind of p/c and check specific semantic relation between policy and credential in some custom-built program since everything in it is free programmable. However, this super capability leads to its undecidability in almost all cases. Most of the subsequent TM approaches, such as DL [11], SD3 [9], Binder [7], and RT_1^C [12], are based on tractable logic programming language DATALOG or its tractable variant $DATALOG^C$. Though tractable in compliance checking, these TM approaches neither include any lexical conceptual structure construction schema, nor support any semantic matching mechanism, so none of them can support the connotation description way. What's worse, the pure DATALOG based ones can't even support the denotation description way because they can't express parameters constraint under the functionless deficiency of DATALOG.

Contribution. In this paper we propose a novel TM approach called OT based on the ontology language OWL 2 EL, which has the following advantages:

(1) OT supports the connotation description way, and partially supports the denotation description way. Local administrators can freely assemble simple abstract concepts from public ontologies into compound concepts to describe parameter constraints, i.e., they can use sharable concept blocks to construct the connotation of parameter constraints like the way they take under human language. As for the capability to describe the denotation of parameter values in p/c, though restricted in some respects, OT shows some merits over RT_1^C —the existing best TM language supporting the denotation description way. (2) The

compliance checking procedure of OT is provable to be tractable. In OT, every access request can be converted into a concept subsumption question under an $\mathcal{EL} + +$ (EL's logic foundation) knowledge base. It is provable in this paper that, the compliance checking procedure of OT is tractable when the concrete data domains used in OT's supporting ontologies are all so called p-admissible domains.

The remainder of this paper is organized as follows. Section 2 presents related work of Trust Management. Section 3 provides the overview of OWL 2 EL and its logic basis. In section 4, we describe the entire OT approach. In section 5 we discuss the efficiency and expressivity of OT. Lastly, section 6 summarizes this paper and presents our future work.

2 Related Work

PolicyMaker is the first TM approach proposed in 1996 by Blaze et al [5]. In PolicyMaker, security policies and credentials are freely programmable. Programmable expression guarantees the expressivity, however the compliance checking procedure of PolicyMaker is polynomial-time solvable in a strictly limited case but undecidable in general [6]. Keynote [4], the second-generator of PolicyMaker, stipulates main parts of p/c to be written as mappings from condition attributes sets to compliance values. Such stipulation makes Keynote more easily to be integrated into application, but meanwhile it lowers the flexibility of expression. Keynote’s undecidability has been proved in many cases [14].

Most of the subsequent TM approaches are based on DATALOG which contributes to the tractability of compliance checking. RT_0 , Binder, and SD3 etc are typical of them. Binder [7] uses the horn clauses to define authorization statements, and arbitrary predicates to represent authorization contents. RT_0 is one sublanguage of RT—a family of Role-based Trust Management languages [13]. RT_0 uses roles to denote the sets of subjects who are granted some specific authorization items and gives the fundamental roles define rules to specify the role membership relation. Except Binder and RT_0 , there are a lot of other DATALOG-based TM approaches, such as Delegation Logic [11], and SD3 etc. The expressiveness of these approaches is constrained as DATALOG is a quite restrictive logic programming language. In fact, they even can’t describe parameter constraints of the authorization terms (such as “ $GPA > 3.6$ ”, “ $institute \in \{A, B, C\}$ ” and so on).

Li et al made use of DATALOG’s extension version— $DATALOG^C$ to design a RT_1^C TM language [11]. By introducing three kinds of parameter constraints: $f = c$, $f \in S$, $f = ref$ (Here, every f is a parameter with specific data type) to modify role types, RT_1^C shows better expression power than former DATALOG-based TM approaches, and still keeps tractable in compliance checking. Actually, RT_1^C is quite adapt to express the extension constraints. For example, it can describe the first authorization item in example 1.2 as “ $student(researchfield \in \{infosec, soft\ engineering, network, \dots\}, institute \in \{A, B, \dots\}, GPA > 3.6)$ ” easily. However, even RT_1^C still can’t express the connotation of parameter constraint such as the ones in the case 1.3.

Cassandra [15] is another TM approach based on $DATALOG^C$. It is able to automatically retrieve missing credentials over the network and supports automated trust negotiation. Furthermore, by introducing the predicates “canActivate”, “canDeactivate”, “canRedCred” etc., Cassandra can make multiple authorisation decisions apart from the one of performing an action, such as activating

and deactivating a role, and requesting a credential. Polakow and Skalka proposed a TM approach based on a monadic linear logic programming language so called LolliMon[16]. Their approach unifies authorization checking and distributed credential retrieval, which solves the distributed certificate chain discovery problem in full RT framework . Though several novel functionalities are added in these TM approaches, their expressiveness about parameter constraints in authorization items is not superior to RT_1^C 's, i.e., they are still unable to express connotation of parameter values.

3 OWL 2 EL

3.1 Syntax and Semantics

An ontology is a formal, explicit specification of a shared conceptualization, including a taxonomy describing the structural concepts of a knowledge area, and axioms capturing the essential rules in that area. OWL 2 Web Ontology Language provides several sub-languages to describe ontologies. One of the profiles of OWL 2 is EL[1] whose logic correspondent is description logic $\mathcal{EL}++$. EL can describe definitions, roles and axioms about different concepts to capture the specific knowledge shared in specific domain. Its description contents form a knowledge ontology about that domain. The syntax of EL can be converted into the one of $\mathcal{EL}++$ to run knowledge reasoning. Table 1 illustrates the abstract syntax of EL and the syntax and semantics of $\mathcal{EL}++$.

In $\mathcal{EL}++$ Atomic concepts (denoted by C,D), atomic roles(denoted by R), features(denoted by f_1, \dots, f_k), and individuals(denoted by a_1, \dots, a_m) can be used to construct complex concepts with constructors. The one through five rows of table 1 illustrate the constructed forms of concepts in $\mathcal{EL}++$. Among them, $p(f_1, \dots, f_k)$ denotes a concrete domain constructor with every f_i in it is a feature which maps every object to an element in that concrete domain. Formally, a concrete domain D is a pair $(\Delta^D, \mathcal{P}^D)$. Δ^D is a data set such as integer set \mathbb{Z} , string set \mathbb{S} and so on, while \mathcal{P}^D is a set of predicate names. Each $p \in \mathcal{P}$ can be interpreted as a set of n-ary tuple $p^D \subseteq (\Delta^D)^n$. The other rows represent the valid concepts axioms which constitute an $\mathcal{EL}++$ knowledge base. The knowledge base is divided into a TBox and an ABox. The TBox comprises four kinds of constraint axioms, which respectively are general concept inclusion axioms(GCIs)— a set of concept inclusion axioms such as $C \sqsubseteq D$, role inclusion axioms (RIs)— axioms for role hierarchy, domain restrictions(DRs) and range restrictions(RRs)—respectively contain axioms for domain and range of roles. ABox is a finite set of individual assertions of two kinds: $C(a)$ or $r(a,b)$. $C(a)$ is an assertion denoting that a is an individual which belongs to the concept C, while $r(a,b)$ denotes there exists a binary relation r between a and b.

The semantics of $\mathcal{EL}++$ is an interpretation $I = (\Delta^I, \cdot^I)$, which consists of a non null abstract individual set Δ^I and an interpretation functions. I maps

¹ EL's current version only supports *ObjectOneof* containing a single individual, though \mathcal{EL} supports nominal with more than one individuals.

Table 1. OWL 2 EL syntax and semantics

Name	OWL 2 EL Syntax	$\mathcal{EL}++$ syntax	Semantics
top	owl:Thing	\top	Δ^I
bottom	owl:Nothing	\perp	\emptyset
nominal	ObjectOneof(a_1, \dots, a_m) ¹	(a_1, \dots, a_n)	$\{x \in \Delta^I \mid x \in \{a_1^I, \dots, a_n^I\}\}$
conjunction	ObjectIntersectionOf(C,D)	$C \sqcap D$	$C^I \cap D^I$
Existential restriction	ObjectSomeValuesFrom(R,C)	$\exists R.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I, (x, y) \in R^I \wedge y \in C^I\}$
Concrete domain	DataSomeValuesFrom(f_1, \dots, f_k, p)	$p(f_1, \dots, f_k)$ for $p \in P^{D_j}$	$\{x \in \Delta^I \mid \exists y_1, \dots, y_k \in \Delta^{D_j}, f_i^I(x) = y_i \text{ for } 1 \leq i \leq k, (y_1, \dots, y_k) \in P^{D_j}\}$
GCI	SubClassOf(C,D)	$C \sqsubseteq D$	$C^I \sqsubseteq D^I$
Equivalent	EquivalentClasses(C,D)	$C=D$	$C^I = D^I$
RI	SubObjectPropertyOf(SubObjectPropertyChain(R_1, \dots, R_k),R)	$R_1 \circ \dots \circ R_k \sqsubseteq R$	$R_1^I \circ \dots \circ R_k^I \subseteq R^I$
Domain	ObjectPropertyDomain(R, C)	$\text{Dom}(R)=C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I, (x, y) \in R^I\} \subseteq C^I$
Range	ObjectPropertyRange(R, C)	$\text{Ran}(R)=C$	$\{y \in \Delta^I \mid \exists x \in \Delta^I, (x, y) \in R^I\} \subseteq C^I$
Concept assertion	ClassAssertion(a, C)	$C(a)$	$a^I \in C^I$
Role assertion	ObjectPropertyAssertion(R,a,b)	$R(a,b)$	$(a^I, b^I) \in R^I$

each concept A to a subset of Δ^I , each role r to a binary relation r^I in Δ^I , each individual a to a a^I in Δ^I and each feature name f to a partial function $f^I: \Delta^I \rightarrow \Delta^D$.

3.2 Knowledge Base Reasoning

$\mathcal{EL}++$ enables a lot of questions to be answered under the knowledge base reasoning. Subsumption questions are the most typical questions among them. Since all of other questions can be reduced to the subsumption questions in $\mathcal{EL}++$, the computational complexity of knowledge base reasoning in $\mathcal{EL}++$ depends on the performance of reasoning about the subsumption problems. Under the semantics of $\mathcal{EL}++$, given two concept C and D, C is subsumed by D under knowledge base K iff $C^I \sqsubseteq D^I$ for every model I of K.

In 2008, Badder et al have proved that subsumption in any knowledge base of $\mathcal{EL}++(D_1, \dots, D_n)$ can be decided in polynomial time w.r.t the knowledge base's size when two conditions are satisfied [3]. The first one is each of D_1, \dots, D_n is p_admissible concrete domain whose definition can be find in [8]. The second condition is that each axiom in RIs of the $\mathcal{EL}++(D_1, \dots, D_n)$ knowledge base should meet the restriction stated below. Before we quote the restriction, we must make this point clear: Under a knowledge base \mathcal{T} , and role names r, s, we write $\mathcal{T} \models r \sqsubseteq s$ iff $r=s$ or \mathcal{T} contains role inclusions:

$$r_1 \sqsubseteq r_2, \dots, r_{n-1} \sqsubseteq r_n, \text{ where } r_1 = r, r_n = s$$

And $\mathcal{T} \models \text{ran}(r) \sqsubseteq C$, iff $\mathcal{T} \models r \sqsubseteq s$, and $\text{ran}(s) \sqsubseteq C \in \mathcal{T}$.

Definition 3.1. *Role Range Inclusion Restriction (RRIR):* for every $r_1 \circ r_2 \dots \circ r_k \sqsubseteq r \in \mathcal{T}$, $n \geq 1$, if $\mathcal{T} \models \text{ran}(r) \sqsubseteq C$, then $\mathcal{T} \models \text{ran}(r_k) \sqsubseteq C$.

4 The OT Approach

4.1 Design of Supporting Ontologies

For every authorization entity, fundamental vocabularies and background knowledge of authorization can be formally conceptualized in three ontologies. The first one is a built-in ontology about the most general taxonomy which is shared by all entities. The second one named Auth-Info ontology defines the taxonomy of parameterized authorization items granted by the entity, while the last domain ontology represents the domain knowledge of the application system which the entity resides in.

Specific Auth-Info ontology and Domain ontology need to be developed by the specific application system administrators. In this paper, we assume every application system can publish its Auth-Info ontology and Domain ontology, or at least their fragments relevant to the public credentials signed by it. Those top-secret application systems which want to hide their authorization architectures and domain vocabularies strictly, such as government or military information systems are not considered in this paper's scope. In the following, we discuss the atomic classes and properties in the three EL ontologies at length.

Built-In Ontology

1. **Principal class:** representing all involved subjects such as access requesters, resource owners, administrators and so on. Every individual of this class is a universal unique ID in the form of URI reference, related to a public Key (for example, based on PKI certification or mapped to ID-Based public key). *Principal* contains two overlapping subclasses: **Grantor** and **Grantee**, which represent the authorization items' grantors and grantees respectively.
2. **Authorization class:** representing various authorization items in different applications.
3. **hasAuth objectProperty:** this object property relates every *Grantee* individual to his/her/its granted *Authorization* individuals.
4. **grantedBy objectProperty:** this connects every *Authorization* to its *Grantor*.

Auth-Info Ontology

1. **subAuthorization classes:** Applications can define and share their own *Authorization* subclasses. Every *subAuthorization* class denote a type of authorizations items and their names can be used as the subject term of those authorization items. For example, some common *subAuthorization* classes are "Role", "Permission", and "Group". The *subAuthorization* classes may form class hierarchies. The application system for CAS referred in example 1.1-1.3 can form a hierarchy of *subAuthorization* classes like the one in Fig.1.
2. **objectProperties:** Every *subAuthorization* class may possess some specific objectProperties to describe its object parameters. Every ObjectProperty *hasOP* has some class in the Domain ontology as its value range. It can be used to express a constraint in the form of $\exists hasOP.E$, where E is a

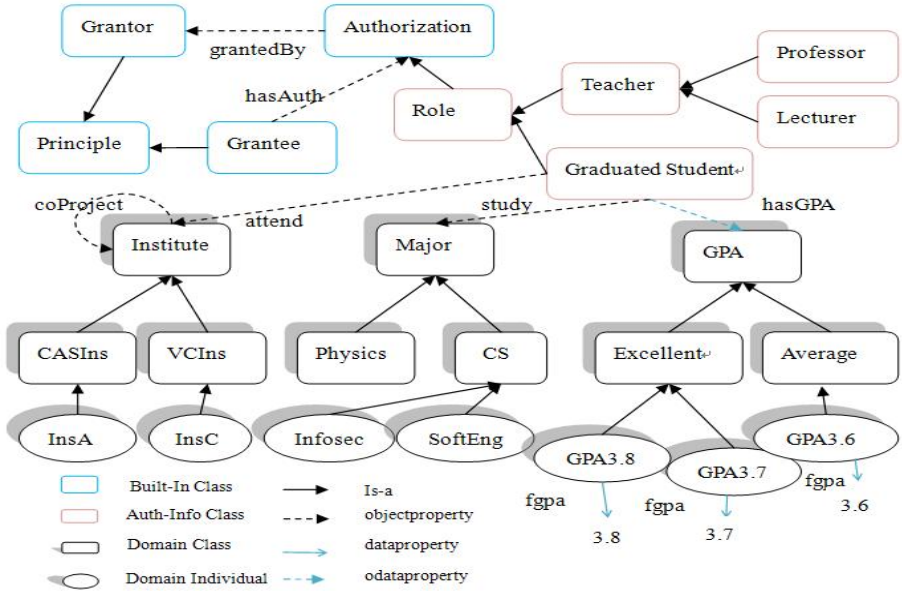


Fig. 1. An example of supporting ontologies in CAS's system

compound class freely constructed from a Domain ontology according to the administrators' needs.

3. **odataProperties:** In order to express constraints over data parameters, such as $age > 40$, $GPA > 3.6$, and so on, dataProperties are needed. However, in real world, even data value has diverse connotations in specific scenarios. For example, the GPA value 3.6 in GAS may stand for a minimum standard for student to be regarded as excellent. Hence all of the dataProperties of subAuthorization classes representing the data parameters are replaced by objectProperties in such way:

For every data parameter dp , we define an objectProperty $hasDP$ in Auth-Info ontology, and define a class DP and a dataProperty fdp for it in Domain ontology, where each value of dp corresponds to an individual of class DP whose feature fdp equals to that value, and $hasDP$'s range is DP . Each constraint about dp can be expressed in the form of $\exists hasDP. (DP[\Box p(fdp)][\Box F])$, where $p(fdp)$ is a data constraint about the parameter such as $< 3.6(fgpa)$, F is a compound class from Domain ontology to explain the connotation of the data parameter's value. In the expression, $\Box F$ and $\Box p(fdp)$ may be absent when administrator doesn't need them. In order to make such design behave well, we assume that every data parameter of authorization item has finite value range. In order to distinguish $hasDP$ from other objectProperties in Auth-Info ontology, we call $hasDP$ s as odataProperties.

Domain Ontology

Domain ontology models knowledge about specific application domain. Many applications have built or will build their own domain ontology no matter whether they use OT or not, such as the digital libraries which use the discipline ontology to classify their papers, and hospitals which use medical ontology to help their digital resource systems operate. The objectProperties of subAuthorization classes reside their ranges in Domain ontology, so administrators can use conceptualized knowledge from Domain ontology to describe the connotation of value constraints of objectProperties. Figure 1 also shows a simple ontology example from the application domain of CAS.

Applications can share the same Auth-Info and domain ontologies with others, or construct their own ones. All of these Auth-Info and domain ontologies should meet the RRIR restriction of definition 3.1 for all of their RI axioms. It can be easily achieved by making sure every R_k 's range is subsumed by R's range when RI axioms are designed.

4.2 Basic Blocks of Credentials and Policies

Based on these ontologies, we can construct the OT credentials and policies and their basic building blocks with abundant vocabularies. There are many ways to format the policies/credentials, such as encapsulating every part of them into specific xml tag, expressing them as well-defined rules and so on. In order to simplify the following presentation, we developed an $\mathcal{EL}++$ -EL mixed syntax to describe OT policies/credentials and their supporting blocks, which also makes the description easy to be converted into the Description Logic level in the process of compliance checking. At first, we use EBNF to define the syntax about the basic building blocks:

$$\begin{aligned} \mathbf{CPR}:: &= ' \exists hasAuth.(' CAI ' \sqcap \exists grantedBy.(' CPR|APR'))' \\ \mathbf{CAI}:: &= subAuth\{ '\sqcap' opconstraint\} \{ '\sqcap' dpconstraint\} \\ opconstraint &::= ' \exists hasOP ' . ' E | (' \{ 's \{ ' , s \} ' \})' \\ dpconstraint &::= ' \exists hasDP ' . (' DP ['\sqcap' p' (' f')] ['\sqcap' F] ')' \\ \mathbf{APR}:: &= ' \{ ' anygrantor ' \}' \end{aligned}$$

Here, subAuth is a subAuthorization class in an Auth-Info ontology, while *hasOP* represents objectProperty and *hasDP* represents odataProperty of subAuth. *E*, *F* are compound classes whose components come from a Domain ontology. *s* is a Domain ontology's individual. *DP* and *f* are class and feature name of the domain ontology. anygrantor is an individual of Class *Grantor*. The former non-standard terms are all presented under the owl 2 EL syntax, while *p* is a predicate stated in $\mathcal{EL}++$ syntax.

A single **Compound Auth-Info expression(CAI)** can be regarded as an authorization item. It represents a group of *Authorization* individuals of the same subAuth class which match all *opconstraint* and *dpconstraint*. In the following, we demonstrate how the authorization items of example 1.3 can be directly interpreted by CAIs in the concept assembling way:

Example 4.1. The CAS authorization item in example 1.3:

$$CAI_{exp1} = graduatedstudent \sqcap \exists study.CS \sqcap \exists attend.ObjectIntersectOf(CASIns, ObjectSomeValueFrom(coProject, VCIns)) \sqcap \exists hasGPA.(GPA \sqcap Excellent)$$

Here, the relevant classes and properties are all from the ontologies in Fig.1.

Example 4.2. MOE authorization item in example 1.3

$$CAI_{exp2} = Applicance \sqcap \exists Type.exStudentToVC \sqcap \exists hasSCH.(SCH \sqcap = 5000(fsch))$$

Here the *Applicance* is a subAuthorization class in Auth-Info ontology of MOE's application system. *Type* is its objectProperty. Together with *SCH* and *fsch* defined in Domain ontology, its odataProperty *hasSCH* is used for describe the data value constraint of "scholarship" parameter. *exStudentToVC* is a domain class representing a category of applications.

Building block **Atomic Principal expression (APR)** is a nominal class representing a single *Grantor* individual, while **Compound Principal expression (CPR)** describes a group of *Grantor* individuals who are granted the authorization item described by CAI and their grantor may be an APR individual or a member of another CPR class. One example of CPR is:

Example 4.3. the principals who are granted the network administrator Role by a super administrator

$$\exists hasAuth.(NetAdmin \sqcap \exists grantedBy.\{superAdmin\})$$

4.3 Complete Expression of Credentials and Policies

Credential: every credential is of the form:

$$\langle Head : Reference \rangle \langle Body : Assertion \rangle \langle Signature \rangle$$

The head part includes all the URI references of relevant signed ontologies and the public Key to verify them. The signature part contains the signature on this credential (the issuer's public Key certification can be included in this part). The authorization assertion in the body is one of the two forms: $AtoA^C$ which provides an authorization to a group of grantees who have gotten other designated authorizations except identities, and $AtoI^C$ which grants the authorization to a grantee with the designated ID.

$$AtoA^C ::= CPR\{\sqcap' CPR\}' \sqsubseteq \exists hasAuth.('CAI' \sqcap \exists grantedBy.\{anygrantor'\})'$$

$$AtoI^C ::= o' : \exists hasAuth.('CAI' \sqcap \exists grantedBy.\{anygrantor'\})'$$

In $AtoA^C$, the "anygrantor" should be the same principal as the issuer of the credential. Otherwise, the credential will be verified as invalid. In $AtoI^C$, o is a grantee individual, and the "anygrantor" should be the same principal as the issuer of the credential.

Policies: Every policy is an authorization assertion stored in local system. In general, policies assert in what condition the local authorizations can be granted to the principals. They are not needed to be signed, connected to some local ontologies, and of the following two forms :

$$\begin{aligned} AtoA^P &::= CPR \{ '\sqcap' CPR \}' \sqsubseteq \exists hasAuth. ('CAI' \sqcap \exists grantedBy. \{ 'localAdmin' \})' \\ AtoI^P &::= o' : \exists hasAuth. ('CAI' \sqcap \exists grantedBy. \{ 'localAdmin' \})' \end{aligned}$$

Here *Grantor* individual localAdmin represents some local administrator.

Add standard term GROUP as macro: Every $\exists hasOP.E$ or $\exists hasDP.G$ ($G = DP[\sqcap p(f)][\sqcap F]$) in a CAI portrays a parameter constraint satisfied by the group of authorizations referred by the CAI. Since the quantifier used in the structures is \exists , the real meaning is “*there exists at least one individual pertaining to E/G that can be ‘hasOPed’/ ‘hasDPed’ by every one of these Authorization individuals*”. If creators want to express “*every individual pertaining to E/G can be ‘hasOPed’/ ‘hasDPed’ by every one of these Authorization individuals*”, they should look up every individual e_i in E/G from the Domain ontology, and write the feature of the CAI as:

$$EOP : \exists hasP. \{ e_1 \} \sqcap \dots \exists hasP. \{ e_m \}$$

Here, hasP represents hasOP or hasDP. To write the feature in this form is a heavy task. So we introduce a GROUP standard term to solve the problem. In this case, grantors only need to write the feature as the following SOP form in their credentials or policies.

$$SOP : \exists hasP. (E/G \sqcap GROUP)$$

When a request comes, credentials and policies are submitted to the decision module of the resource provider’s OT subsystem. OT subsystem will recognize every GROUP standard term and automatically look up the corresponding Domain ontology to replace every SOP structure with the EOP structure.

4.4 Compliance Checking

In OT, every access request is made by one user, and decision is made under lots of relevant credentials and policies. Apparently, users are responsible for keeping, delegating or submitting their own $AtoI^C$ credentials. However, who can be responsible for collecting the relevant $AtoA^C$ credentials for every request? We suggest two reasonable and feasible strategies on this: (1) With a **free-style strategy**, resource providers don’t control the delegation depth of authorization. So the middle level $AtoA^C$ credentials are free issued out of their control and the requesters should be responsible for gathering them and submitting to the OT subsystem. (2) In the **strick-control strategy**, providers will designate explicitly all the $AtoA^C$ credentials trusted by themselves, but permit the issuer to sign $AtoI^C$ credentials freely. Henceforth all the $AtoA^C$ credentials are stored in the OT subsystem of the resource provider. Then requesters only need to submit their own $AtoI^C$ credentials along with the request.

Under any one of the two strategies, OT subsystem is supposed to get enough credential sets and can check the compliance efficiently. When an access request is submitted, the compliance checking procedure is processed in two phases.

Preparation phase: After receiving the request, server need to (1) collect relevant credentials according to free-style or strick-control strategy, (2) verify every

credential collected. If any credential is invalid drop it, (3) retrieve the relevant ontologies of every credential and verify their integrity. If any ontology fails in this step, drop the corresponding credential.

Execute phase: (1)Expand every GROUP macro appeared in collected credentials. (2)Merge and translate the relevant ontologies, authorization assertions in local policies and credentials into a synthesized $\mathcal{EL}++$ knowledge base TB, and translate the request into a subsumption problem “ $\{req\} \sqsubseteq \exists hasAuth.(CAI \sqcap \exists grantedBy.\{localAdmin\})?$ ” where ‘req’ is the requester individual’s ID, and CAI is the requested local authorization. (3)Run the reasoner and check if the subsumption holds under TB.

Next, we present a running example about the compliance checking. Supposed that the example 1.3 denotes a local policy of MOE, it will be expressed as such a OT policy:

$$\begin{aligned} &\exists hasAuth.(CAI_{exp1} \sqcap \exists grantedBy.\{CAS\}) \\ &\sqsubseteq \exists hasAuth.(CAI_{exp2} \sqcap \exists grantedBy.\{MOE\}) \end{aligned}$$

Meanwhile, Alice has a credential issued by Institute A of CAS like this:

$$\begin{aligned} Alice : &\exists hasAuth.(graduatedStudent \sqcap \exists study.\{infoSec\} \sqcap \exists attend.\{InsA\} \sqcap \\ &\exists hasGPA.\{GPA \sqcap = 3.8(fgpa)\} \sqcap \exists grantedBy.\{CAS\}). \end{aligned}$$

If she wants to apply for the scholarship, she can make the request and submit her student credential MOE’s OT subsystem. Then after relevant ontologies and credentials verified to be valid, MOE will generate a subsumption question: $\{Alice\} \sqsubseteq \exists hasAuth.(CAI_{exp2} \sqcap \exists grantedBy.\{MOE\})$. Under the relevant ontologies, MOE can infer that *InsA* is an individual of “*CASIns*” and is cooperating with the VCIns *InsC*, and *infoSec* is one research field in *CS*. No doubt that finally this subsumption question can be reasoned to be true. So Alice is a lucky girl to be qualified for this application.

5 Property Analysis

So far, we have presented the whole design of OT approach. As follows, we will present the efficiency result about OT’s compliance checking procedure, and then give a detailed comparison between OT and former TM approaches in expressivity.

5.1 Compliance Checking Efficiency

As referenced at section 4.4, the compliance checking procedure is divided into two phases. Apparently, the preparation phase won’t cost much time. Hence, the efficiency of compliance checking depends on the execute phase. Before introducing our conclusion about the efficiency of execute phase, we quote the p_admissible concrete domain’s definition [8].

Definition 5.1. A concrete domain D is p -admissible if:

1. the satisfiability of $\bigwedge_i p_i(f_{i,1}, \dots, f_{i,n_i})$ and implication of $\bigwedge_i p_i(f_{i,1}, \dots, f_{i,n_i}) \rightarrow q(f_1, \dots, f_n)$ in D are decidable in polynomial time;
2. D is convex, i.e., if a conjunction of atoms of the form $p(f_1, \dots, f_k)$ implies a disjunction of such atoms, then it also implies at least one of the disjuncts.

Theorem 5.1. If every concrete domain D_i used in the $\mathcal{EL}++$ knowledge base TB synthesized in execution phase is p -admissible, then the execution phase can be finished in polynomial time w.r.t the size of TB .

Proof. Every RI axiom of every Domain ontology or Auth-info ontology meets RRIR when they are designed, so RRIR is kept for TB . Besides, every concrete domain D_i in TB is p -admissible. Thus, TB satisfies both the p -admissible condition and the RRIR condition. According to the discussion in section 3, the subsumption problem under TB is tractable w.r.t the size of TB . Because the Domain ontology of every relevant credential is part of TB 's sources, the subsumption problem under the Domain ontology's corresponding $\mathcal{EL}++$ knowledge base is tractable w.r.t n if the size of that Domain ontology is n too.

Now we consider all steps in execution phase. In the first step, for every credential which has SOP expression to expand, the only time-consuming job is to find out all the individuals belonging to E_i of the SOP in its Domain ontology. This job can be done in polynomial time w.r.t the size of the Domain ontology n , because it needs no more than n subsumption decisions and every decision is solvable in polynomial time w.r.t n . Then the sum of all credentials' expansion time is polynomial w.r.t the size of the whole TB . The time spent in the second step can be neglected, while the third step is a polynomial time solvable subsumption decision in TB , so the whole execution phase can be finished in polynomial time w.r.t the size of TB . \square

There exist a lot of p -admissible concrete domains, a typical one of which is $D = (Q, \mathcal{P})$, where Q is a domain of real numbers, and \mathcal{P} consists of unary predicates $=, >$ for every q in Q . Though other non p -admissible may introduce intractability, they can still maintain the decidability if the satisfiability and implication in them are decidable. So the efficiency of OT can be kept at the decidability level at most cases.

5.2 Expressivity

In expressivity, with the unique concept assembling way to describe parameter constraints, OT can support not only the connotation description but also the denotation description.

(1) connotation description

As showed in the former sections, OT provides a desirable method for policy administrators to express the connotation of authorization. In OT, administrators can use concepts from external entity's Auth-Info ontology and Domain ontology

to express the connotation of authorization item they need. Since the ontologies are under the dynamic and lasting maintenance of external administrators, they can help the compliance checking procedure to automatically find out the current valid value sets for every property's connotation. But in the former efficient TM approaches, these can't be achieved.

(2) denotation description

To our best knowledge, RT_1^C is the best solution when describing denotation of parameter constraints. Thus we just make a comparison between RT_1^C and OT about their denotation description capabilities in the following:

Due to the absence of variable in OWL 2 EL, the third kind of parameter constraint " $f = ref$ " of RT_1^C is problematic in OT. However, OT can express RT_1^C 's " $f \in S$ " (" $f=c$ " is just one kind of " $f \in S$ ") parameter constraints in its own way. Without loss of generality, we assume there is an authorization item expressed as $R(f \in S)$ in RT_1^C . In general, there are two translation versions about it in OT:

a. $R(f \in S)$ appears in the body part of any RT_1^C rule(e.g. $A.R_1(..) \leftarrow B.R(f \in S)$): since it's in the rule's body, f 's hidden quantifier is \exists . If S belongs to a linearly ordered range set, we can use $R \sqcap \exists hasDP.(DP \sqcap \in S(f))$ to represent $R(f \in S)$. When the S belongs to some set of unordered enumeration elements, $R \sqcap \exists hasOP/hasDP.\{s_1, \dots, s_n\}$ can be used where every s_i corresponds to an elements in S .

b. $R(f \in S)$ appears in the head part of any RT_1^C rule(e.g. $A.R(f \in S) \leftarrow B.R_1$): Since it's in the rule's head, f 's hidden quantifier is \forall . So we can use $R \sqcap \exists hasDP.(DP \sqcap \in S(f) \sqcap GROUP)$, or $R \sqcap \exists hasOP/hasDP.\{s_1, \dots, s_n\} \sqcap GROUP$ to represent $R(f \in S)$.

In detail, these translation ways show both disadvantages and advantages:

(1) disadvantages: The introduction of predicate " $\in S$ " will cause intractability since domain $D=(C, \{\in S\})$ where C is any linearly ordered set is non p-admissible (Despite this fact, the decidability can still be maintained). On the other hand, a few datatypes such as long, double are not supported in the current EL version, which causes those S sets of these restricted datatypes can't be expressed at present.

(2) advantages: As discussed before, the hidden quantifier for f in $R(f \in S)$ is decided by the position of $R(f \in S)$ in a rule. That is to say, $R(f \in S)$ can become neither $\exists f R(f \in S)$ in head nor $\forall f R(f \in S)$ in body. This inability hinders it to express some human thoughts about authorization. But in OT, GROUP macro can show up in head or body of rules, so this problem doesn't exist for OT.

6 Conclusions

In this paper, we proposed an OWL 2 EL based TM approach OT, illustrated OT's capability in expressing connotation of TM authorization policies and proved OT's tractability in compliance checking. In the future, we will study how to extend OT's expressivity further, such as introducing variables into OT's policy language.

References

1. OWL EL Introduction. See http://www.w3.org/TR/owl2-profiles/#OWL_2_EL.
2. Baader,F.,Brandt,S., Lutz,C.: Pushing the \mathcal{EL} envelope. In: Proceedings of IJCAI (2005) 364–369.
3. Baader,F., Brandt,S., C. Lutz.: Pushing the \mathcal{EL} envelope further. In: Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions. (2008).
4. Blaze, M., Feigenbaum,J., Keromytis,A.D.: Keynote: Trust management for public-key infrastructures (position paper). In: Proceedings of the 6th International Workshop on Security Protocols. (1999) 59–63.
5. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. In: Proceedings of the 17th Symposium on Security and Privacy. (1996) 164C-173
6. Blaze, M., Feigenbaum, J., Strauss,M.: Compliance checking in the policymaker trust management system. In: Proceedings of the Second International Conference on Financial Cryptography. (1998) 254–274,.
7. DeTreville,J.: Binder, a logic-based security language. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. (2002)page 105–113.
8. Baader,F., Lutz, C.: Pushing the \mathcal{EL} envelope. Technical report, LTCS-Report ltcs-05-01, Inst.for Theoretical Computer Science, TU Dresden, See <http://lat.inf.tudresden.de/research/reports.html> (2005).
9. Jim,T.: SD3: A trust management system with certified evaluation. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy. (2001) 106–115.
10. Li,N.: Local names in SPKI/SDSI. In: Proceedings of the 13th IEEE workshop on Computer Security Foundations. (2000) 2–15.
11. Li,N., Grosof, B.N., and Feigenbaum,J.: Delegation logic: A logic-based approach to distributed authorization. ACM Trans. Inf. Syst. Secur., 6(1):128–171 (2003).
12. Li,N., Mitchell,J.C.: Datalog with constraints: A foundation for trust management languages. In: Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages. (2003).
13. Li,N., Mitchell,J.C., Winsborough,W.H.: Design of a role-based trust-management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. (2002) 114-130.
14. Li,N., Mitchell,J.C., Winsborough,W.H.: Beyond proof-of-compliance: security analysis in trust management. J. ACM, 52(3):474–514, 2005.
15. Becker,M.Y., Sewell,P.: Cassandra: flexible trust management and its application to electronic health records.In IEEE Computer Security Foundations Workshop.(2004)139-154
16. Polakow,J., Skalka,C.:Specifying Distributed Trust Mngement in LolliMon. In Proceedings of the 2006 workshop on Programming languages and analysis for security (2006)37–46.

A User Trust-Based Collaborative Filtering Recommendation Algorithm*

Fuzhi Zhang, Long Bai, and Feng Gao

School of Information Science and Engineering Yanshan University,
Qinhuangdao , 066004, Hebei Province, P. R. China

Abstract. Due to the open nature of collaborative recommender systems, they can not effectively prevent malicious users from injecting fake profile data into the ratings database, which can significantly bias the system's output. With this problem in mind, in this paper we introduce the social trust of the users into the recommender system and build the trust relation between them. The values of trust among users are adjusted by using the reinforcement learning algorithm. On the basis of this, a user trust-based collaborative filtering recommendation algorithm is proposed. It uses the combined similarity to generate recommendation, which considers not only the similarity between user profiles but user trust as well. Experimental results show that the proposed algorithm outperforms the traditional user-based and item-based collaborative filtering algorithm in recommendation accuracy, especially in the face of malicious profile injection attacks.

Keywords: collaborative filtering; recommender system; trust model; malicious attack; Reinforcement learning.

1 Introduction

At present, personalized collaborative recommender systems have become an important part of many e-commerce Web sites. However, such recommender systems introduce security issues that must be solved if users are to perceive these systems as objective, unbiased, and accurate [1]. The open nature of collaborative recommender systems provides an opportunity for malicious users to access the systems with multiple fictitious identities and insert a number of fake user profiles in an attempt to bias the recommender systems in their favor. Traditional collaborative recommender systems can not prevent this kind of malicious attack. Thus how to ensure the quality of recommendations for personalized collaborative recommender systems in the face of profile injection attacks has become an important issue.

Recent research on the security issues of collaborative recommender systems has focused on techniques that can be used to protect the predictive integrity

* This work was supported in part by the National Basic Research Program of China (No.2005CB321902), and the Natural Science Foundation of Hebei Province, China (No.F2008000877).

of collaborative recommenders from malicious profile injection attacks. Research work falls into two categories: techniques for detecting and discounting biased profiles [2,3]; and techniques that increase the robustness of the recommender systems [4,5]. In this paper, we will explore to combine the user trust mechanism with collaborative filtering algorithm for the purpose of improving the robustness of collaborative recommendation algorithm and ensuring the quality of recommendations.

Montaner et al. [6] introduced a trust model into the recommendation algorithm, so users could get the recommendations from the trust-building group. The tentative idea was that trust-factor was based on the customer's satisfaction with the recommended items and trust value could be dynamically adjusted. It was a fresh idea for the recommendation algorithm. The drawback of this method was lack of trust information among users at the beginning of recommendation, and what's more it was inefficient to build the trust group. So it was not an effective way to defend against the malicious noise.

Massa et al [7] proposed a method that users who accepted the recommendations would evaluate the recommended items. The active user would get a rating that stands for the trust value of target user to the active user. The trust information was propagated among users who had a trust relation with the accepter. In this way a relation network with the trust value among users in the group would be built, even if they didn't have a direct interaction with each other. This was an effective way to build the trust network among users. However, due to the lack of restriction on the propagation of trust among users in the group, this method might lead to the flooding of trust and cause it to be out of control in the end. In addition, the authors did not give an effective way to measure the relation between users, so the initial trust values could not be effectively quantified, which could not ensure the reliability of trust values.

John O'Donovan et al [8] proposed an approach to overcome the shortcoming mentioned above. The basic idea was to build a relation between users with recommended items. Based on the tentative idea, there would be a higher weight to active user who had more accurate recommendations on items than those with poor records within the recommendation process. They supposed that users with a high authentic value have less intention to deceive others. The item-trust recommendation algorithms were more effective to defend the random attacks [9], but if the malicious users changed the attack strategies, in particular, they had some collaboration with others; this method would not effectively cut down the negative effect. Due to the lack of trust between users, they couldn't clearly judge who accepted item, who can be trusted or not.

To overcome the drawback, in this paper we explore to exploit trust information explicitly expressed by the users to improve the robustness of recommender systems. We give a user trust model and build a trust network for users by reinforcing learning. The trained items are chosen from the items which have been rated by the users. We also propose a user trust-based collaborative recommendation algorithm to defend malicious noise. The experimental results show that our algorithm has a significant improvement in stability compared with the standard

collaborative filtering algorithm, and the algorithms with the constraint of user trust are more robust than other model-based algorithms, especially for counter-acting malicious noise.

2 Background

2.1 Traditional Collaborative Recommendation Algorithm

Traditional collaborative recommendation algorithm, for example, the user-based collaborative filtering, uses the similarity of user profiles to form a neighborhood of peer users with similar tastes, then extrapolates the user’s predicted rating for a target item from the ratings of his or her peer users [1]. The core of this algorithm is to compute the similarity between users. There are several methods can be used to compute the similarity of users, such as cosine correlation coefficient, modify cosine similarity, and Pearson correlation coefficient. In this paper we use the Pearson correlation to calculate the similarity of users.

Let $D = \{U, I, R\}$ be a data source of a recommender system, where $U = \{user_1, user_2, \dots, user_m\}$ is a set of users of the system, $I = \{item_1, item_2, \dots, item_n\}$ is a set of items of the system, and R is a user ratings matrix, where $r_{i,j} \in R$ represents the rating of $user_i$ on $item_i$. The similarity between $user u$ and $user n$ is given by the following Pearson’s correlation coefficient Equation [10]:

$$Sim(u, n) = \frac{\sum_{C \in I_{u,n}} (R_{u,c} - \overline{R_u})(R_{n,c} - \overline{R_n})}{\sqrt{\sum_{C \in I_{u,n}} (R_{u,c} - \overline{R_u})^2} \sqrt{\sum_{C \in I_{u,n}} (R_{n,c} - \overline{R_n})^2}} \tag{1}$$

Where $R_{u,c}$ and $R_{n,c}$ are the rating of $user u$ and $user n$ on $item c$, $\overline{R_u}$ and $\overline{R_n}$ are the average ratings over all rated items for u and v , respectively. The set $I_{u,n}$ stand for the rating items on which $user u$ and $user n$ have co-rated. It is important to underline that the coefficient can be computed only if there are items rated by both the users.

2.2 Reinforcement Learning

Reinforcement learning is a machine learning method to solve problem through trial-and-error interactions with a dynamic environment. In the standard reinforcement learning model, an agent was connected to its environment via perception and action. In this paper we use the idea of the reinforcement learning to build the direct trust between users. Formally, the reinforcement learning model consists of [11]:

- (1) a set of environment states: S ;
- (2) a discrete set of agent actions: A ;
- (3) a reward function $R : S \times A \rightarrow R$;
- (4) a state transition function $T : S \times A \rightarrow \prod(s)$, where a member of $\prod(s)$ is a probability distribution over the set S . We write $T(s, a, s')$ for the probability of making a transition from state s to s' using action a .

2.3 Item-Level Trust

In [8] the authors proposed a view that the users who had made lots of accurate recommendation predictions in the past could be viewed as trustworthy compared with those made many poor predictions. It is a good manner to evaluate the recommendation quality by the target user. The evaluation can be viewed as a trust value to the active user. In traditional collaborative filtering, participants were viewed as collectivity to predict the items for target user, which is hard for assessor to estimate trust for each user.

Accordingly, we separately calculate correctness of producer's prediction by comparing predicted rating and the actual rating of the target users.

Let $T_n(i, u)$ be a trust value of user u for user n , if user n predicts the rating for the user u , the trust value is given by Equation 2.

$$T_n(i, u) = 1 - v_n^i \quad (2)$$

Where variable v_n^i is the deviation factor which represents the deviation degree of the active user n to the target user u on item i . The value of v_n^i is given by Equation 3 [12].

$$v_n^i = \frac{|r_n^i - r_u^i|}{d} \quad (3)$$

Where variable r_n^i is the rating of user u on item i , r_u^i is the rating of the target user u on item i , d is the span value of item's rating and the default value of the variable is 5.

3 User Trust Model and Generation Algorithm

Traditional user-based collaborative recommendation algorithm uses the similarity of users' tastes to generate recommendations. This profile-level similarity method is subject to manipulation by malicious users. Thus the reliability of users should take into account within the recommendation process. In this paper we use trust between users to express the reliability of users and combine user trust with user-based collaborative recommendation algorithm.

3.1 Definition of User Trust Model

In this section we introduce a formal trust relationship which is the extension of the representation used in [8]. To model the degree of trust, we assume that target user can assign a certain value to the active user by using the co-rated items of the users.

Trust metrics can be imported to help the target user to compute the trust value about the active users. The trust metrics is computable on most users, even on pairs of users who have only one co-rated item. A user is also able to establish trust via trust propagation on users with whom has no co-rated item. We use two types of trust: direct trust and recommendation trust. The former can be constructed

by users with exchange experiences such as friendship, good views. The latter is credit of a user award by the other users who are reliable by public.

Definition 1. Direct trust:

Let T_n^u represent the direct trust of the target user for the active user, the direct trust value is given by Equation 4.

$$T_n^u = \frac{\sum_{i=1}^k t_n^i}{\sum_{i=1}^k |t_n^i|} \tag{4}$$

Where t_n^i represents the trust value of target user u for active user n on item i . The trust value can be divided into five parts, $t_n^i \in [-0.2, -0.1, 0, 0.1, 0.2]$, the value of each part depends on the deviation factor of v_n^i which is given by equation 3, and k is the number of the set which contains items that the active user and the target user have co-rated.

The direct trust has the following property. If the active user has a positive experience to influence the target user, then the positive experience will be strengthening. If there has a negative experience, it will be given a distrust value as a punishment. In a real recommender system, however, the number of items is huge and the number of ratings provided by each user is very small, so there are few co-rated items between users. In such cases the traditional collaborative recommendation algorithm can't effectively calculate the similarity between users, which provides few chances for users to interact with each other. As a result, the direct trust is also scarce. Thus the single direct trust is not enough to express the trust relation between users. We need to introduce the recommendation trust into the trust building process.

Definition 2. Recommendation trust:

The recommendation trust is computed by target user's trust group who has an interaction with the active user. Let m be a set of trust group of the target user, which contains the users who have a reliable intercourse with the target user u . Let w_i be a trust factor of the target user for the active user, which depends on the history of intercourse experiences. Let T_m^n represent the recommendation trust that is computed by the set of user m who has a direct trust with active user n .

$$T_m^n = \frac{\sum_{i=1}^k w_i T_{m_i}^n}{\sum_{i=1}^k w_i} \tag{5}$$

Where w_i is a trust value of the target user u for the reliable user in the set m , $T_{m_i}^n$ is the direct trust value of user m_i (in the set m) for the active user n .

Using Equation 5, we can build a trust relation between target user and active user with the expression of the trust set that has a direct interaction with the target user. In this way we can reduce the negative influence caused by lacking information about the active user. If a user has a poor credit record, the record about the user will spread via recommendation trust, so the user will have less chance to participate in the recommendation. On the contrary if a user has a perfect record, he will be viewed as a genuine user instead of a malicious.

Definition 3. User trust value:

Let $Trust_n^u$ stand for the trust value of the target user u for the active user n , it is combined with the direct trust T_n^u , which is the value of the target user u for the active user n , and the recommendation trust which is the expression of the set trusted by the target user u . The combined trust value is given by Equation 6.

$$Trust_n^u = \alpha T_n^u + \beta T_m^n \quad (6)$$

Where α, β are weighting factors to adjust the two parts, they are constrained by the equation $\alpha + \beta = 1$. The parameter α is used to balance the direct trust and β is used to balance the recommendation trust. If users have more successful interaction, there will be more similarity and direct trust between those users who have interaction with each other.

As mentioned above, a user usually rates on a very small part of the items. It is difficult to compute direct trust between arbitrary users. We can introduce recommendation trust to solve this problem. The proportion of recommendation trust must be restricted, because it will bring about a distortion recommendation. For example, a target user 'A' may have little direct trust with the active user 'D' due to the lack of co-rating items. Maybe the two users have different interest. If user 'D' has similar tastes with group 'm' trusted by user 'A', and they have a good trust with each other. Thus the recommendation trust of T_m^B will be a good value. As a result, the active user 'D' who has little common interesting with 'A' will be a candidate to participate in the prediction. We can introduce the parameters to balance the direct trust and recommendation trust.

The user trust includes two parts. The first part is the interaction of target user and active user. If the active user has many good predictions for target user, he or she will not deceive the target user. The second part is the opinions of the other users who are trusted by the active user, they are also an important reference to the target user. In this way the target user will receive a credible recommendation from the producers.

3.2 User Trust Generation Algorithm

In the open collaborative recommendation environment, ensuring the reliability of users' ratings is very important to improve the quality of recommendation. With this problem in mind, we introduce the user trust into the collaborative filtering algorithm and calculate the degree of user trust using the idea of reinforcement learning.

In this section, we will provide three algorithms. The first algorithm is to calculate the direct trust between the target user and the active user, the second

is to compute the recommendation trust between users, and the third is to calculate the combined trust of direct trust and recommendation trust.

Algorithm 1

Input: target user, active user

Output: the value of the direct trust

Step:

1: **begin**

2: **repeat**

3: $ItemU \leftarrow getItem(userId)$

/*To store the rating items of target user into set ItemU*/

4: **end repeat**

5: **repeat**

6: $CommItem \leftarrow getCommonItem(ItemU, userID)$

/*To get the co-rated items by target user and active user*/

7: **for** $i = 1$ to k **do** /* k is the number of co-rated items*/

8: compute the trust value about the common item i , by Equation 3

9: **end for**

10: compute the direct trust value T_u^n by Equation 4

11: $Trust_Array1 \leftarrow T_u^n$ /*To store the direct trust value into the matrix*/

12: **end repeat**

13: **end**

The function of this algorithm is to build trust relationship between the target user and the active user. Line 2 to 4 is to collect the co-rated items for the preparation of computing the direct trust value, and the line 5 to 12 is the core of building the direct trust. When calculating the correctness of active user's recommendation, we separately perform the recommendation process by using active user as the target user's sole recommendation partner.

The recommendation trust is to get the expressions about the active user from the target users' partners. For this purpose, the trust derivation algorithm is used to derive all possible direct trust relationships with the active user as trusted factor from a given set of initial trust relationships. The algorithm tries for each recommendation trust expression to derive as many new trust expressions as possible. Then the considered recommendation trust is removed from the set and the new recommendation trusts are inserted into it.

Algorithm 2

Input: target user, active user, the number of the target user's partner k , and the trust matrix $Trust_Array1$

Output: the value of the recommendation trust T_m^n

Step:

1: **begin**

2: **repeat**

3: $Trust_List \leftarrow query(u, Trust_Array1)$

/*To get the trust set of the target user*/

```

4: end repeat
5:  $Trust\_quene(key, w_i) \leftarrow Sort(Trust\_List, k)$ 
   /*To obtain the k nearest of the target user's neighbor
   that is order by the direct trust between users. The
   variable key is user's id, while the  $w_i$  is the value of
   the trust.*/
6: for i =1 to k do
7:    $T(m_i, n, T_{m_i}^n) \leftarrow indexOf(key, n, Trust\_Array1)$ 
   /*To index the direct trust  $T_{m_i}^n$  between the user  $m_i$  and active user  $n^*$ */
8: end for
9: Compute the recommendation trust  $T_m^n$  by Equation 5
10: end

```

This algorithm consists of two parts. The first part, from line 2 to 4, is to get the trust set of the target user and prepare for computing the recommendation trust; the second part, from line 5 to 9, is to compute the recommendation trust of active user's.

Algorithm 3

Input: target user, active user

Output: the value of the combination trust $Trust_n^u$

Step:

```

1: begin
2:    $T_n^u \leftarrow T(u, n_j)$ 
3:    $T_m^n \leftarrow T(u, m)$ 
4:    $Trust_n^u \leftarrow \alpha T_n^u + \beta T_m^n$ 
5:    $Trust\_Array2 \leftarrow Trust_n^u$ 
6: end

```

This algorithm performs the combination of direct trust and recommendation trust, which is computed by Equation 6. As mentioned earlier, we assume that the value of each recommendation can be measured by the recommendation precision to reflect the reputation of active user, and the reputation value (trust value) will be import in the k-nearest neighbor algorithm.

4 User Trust-Based Collaborative Filtering Algorithm

4.1 Description of Algorithm

With the trust factor introduced into the traditional collaborative recommendation algorithm, there will be two factors to influence the result of prediction. One is the similarity of the users' tastes; the other is the trust value between users, which is the combination of direct trust and recommendation trust.

The steps of the trust-based collaborative recommendation algorithm are as follows.

(1) To generate the trust matrix $Trust_Array2$ of the users

$$Trust_Array2 = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1n} \\ T_{21} & T_{22} & \cdots & T_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n1} & T_{n2} & \cdots & T_{nn} \end{bmatrix}$$

The matrix is an n -order square that filled with trust value of the users, which has been calculated by offline. The element T_{mn} is the combination of direct and recommendation trust, which stands for the relation between the target user m and the active user n . Values symmetrically distributed on the both sides of the matrix diagonal.

(2) To generate the similarity matrix by Equation 1. Users with similar interest or behavior would be gathered (e.g., accessed the same type of information, purchased a similar set of products, liked or disliked a similar set of goods). Sim_{mn} similarity value between user u and n .

$$Sim = \begin{bmatrix} Sim_{11} & Sim_{12} & \cdots & Sim_{1n} \\ Sim_{21} & Sim_{22} & \cdots & Sim_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Sim_{n1} & Sim_{n2} & \cdots & Sim_{nn} \end{bmatrix}$$

(3) To calculate the combined similarity values, the simplest way to combine trust value with similarity to produce a compound weight that can be got by Resnick’s formula. We use the CITEM algorithm introduced in [9] to generate the improved similarity value. This approach is a combination of trust-based weighting of producer’s reputation with interesting similarity of producer profile ratings on the item.

$$W(u, n) = \frac{2(Sim(u, n))(Trust(u, n))}{Sim(u, n) + Trust(u, n)} \tag{7}$$

(4) The weighting approach simply adds the extra metric of trust to the standard similarity weighting in Resnick’s prediction formula. The modified version of Resnick’s formula is shown in Equation 8 to include the trust weighting.

$$pred(u, i) = \overline{R}_u + \frac{\sum_{Neu} W(u, n) * (R_{n,i} - \overline{R}_n)}{\sum_{Neu} (|W(u, n)|)} \tag{8}$$

We use Resnick’s algorithm to compute the prediction of item i for target user u . Where Neu is the set of k similar neighbors that have rated on item i ; $R_{n,i}$ is the rating of i for neighbor n ; \overline{R}_u and \overline{R}_n are the average ratings over all rated items for user u and user v , respectively.

According to the above-mentioned, we give the user trust-based collaborative filtering recommendation algorithm as follow.

Algorithm 4Input: target user, trust matrix *Trust_Array2*

Output: the prediction rating of the item

Step:

```

1: begin
2:   for i=1 to m do /*The 'm' is the number of the neighbor*/
3:     Item  $\leftarrow$  getItem(u)
       /*Get the rating items of the target user*/
4:     for j=i to m do
5:       Comm_Item_Array  $\leftarrow$  getCommonItem()
6:     end for
7:     Comm_List(u,n)  $\leftarrow$  Comm_Item_Array
8:     Sim  $\leftarrow$  Sim(u,n)
9:     Trust(u,n)  $\leftarrow$  Trust_Array2
10:    Compute the similarity  $W(u,n)$  with Equation 7
11:    Sim_List  $\leftarrow$   $W(u,n)$ 
12:  end for
13:  Neu  $\leftarrow$  Sort(Sim_List)
14:  for i=1 to k do
15:    Compute the prediction value  $Pred(u,i)$ 
16:  end for
17:end

```

This algorithm consists of two parts. The first part, from line 2 to 13, is the core of the algorithm. In this part, the first step is to build the trust relation of users using the algorithm 3; the second step is to get the neighbors of the target user with the improved algorithm. The second part, from line 14 to 16, is to predict the ratings of the items by Equation 8.

Using trust metrics in the traditional collaborative recommendation model can enhance the robustness of recommendation algorithm and improve the prediction accuracy of collaborative filtering in the face of malicious noise.

4.2 Algorithm Computational Complexity

The computational complexity of our user trust-based collaborative recommendation algorithm depends on the amount of time required to build the model and the amount of time to compute the recommendation using this model.

To build the model, we need to compute the similarity between each user u and all the other users, and select the most similar users. Let n be the number of users in a recommender system, m the number of items. We need to compute $n(n-1)$ similarities, so the time complexity for similarity computation is $O(n^2)$. Each potentially requires m operations, the time complexity is $O(m)$. According to Pearson's correlation coefficient Equation 1, the time complexity is $O(n^2+m)$. In Algorithm 4, the user trust value used in Equation 7 can be computed via offline, it does not bring about additional computation. Thus our user trust-based collaborative recommendation algorithm does not increase the computational complexity.

5 Experimental Results

To evaluate our algorithm that combined user trust with the collaborative filtering approach. We have carried out our experiments and given a comparison with other collaborative filtering algorithm. The experimental condition: the hardware environment is PC of Intel Pentium IV 2.4GHz CPU and 1G RAM, Operating system is Windows XP Professional. Data stored by Mysql database, and algorithm programming with Java.

In our experiment we used the publicly available dataset provided by MovieLens Site (<http://movielens.umn.edu/>). The site is a Web-based recommendation system provided by GroupLens (<http://www.grouplens.org>) team. The dataset contains 100,000 ratings on 1682 movies by 943 users. All ratings are integer values between 1 and 5, where 1 is the lowest (disliked) and 5 is the highest (most liked). Our data includes all the users who have rated at least 20 movies.

5.1 Evaluation Metrics

The most used technique for evaluating Recommender Systems is based on leave-one-out. It is an offline technique that can be run on a previously acquired dataset and involves hiding one rating and then trying to predict it with a certain algorithm. The predicted rating is then compared with the real rating and the difference in absolute value is the prediction error. The procedure is repeated for all the ratings and an average of all the errors is computed, the Mean Absolute Error (MAE).

MAE is usually applied to measure the accuracy, which measures the average absolute difference between the predictions and the ratings over all items. The formula is as follows:

$$MAE = \frac{\sum_1^N |P_i - Q_i|}{N} \quad (9)$$

Where N is number of the rated items, and P_i is predicted rating on the item i , Q_i is rating of target user on item i .

5.2 Accuracy Analysis

To evaluate the recommendation precision of our algorithm, we have carried out the experiments with our user trust-based collaborative filtering recommendation algorithm, traditional collaborative filtering recommendation algorithm [10] and item-trust collaborative filtering recommendation algorithm [8].

Figure 1 shows the comparison of recommendation quality using different weighting factor α . The performance of our user trust-based collaborative recommendation algorithm is better than that of the user-based collaborative recommendation algorithm, and the different weight value of α performs different accuracy. Appropriate value of the weight can effectively balance direct trust and recommendation trust and avoid the similar users being hijacked by the user trust value, which may lead to a distortion recommendation.

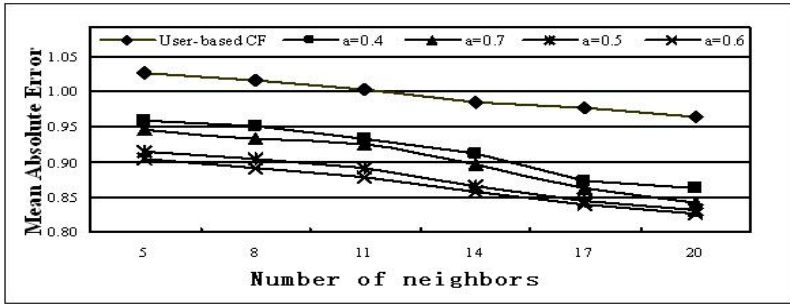


Fig. 1. Comparison of recommendation quality using different weighting factor

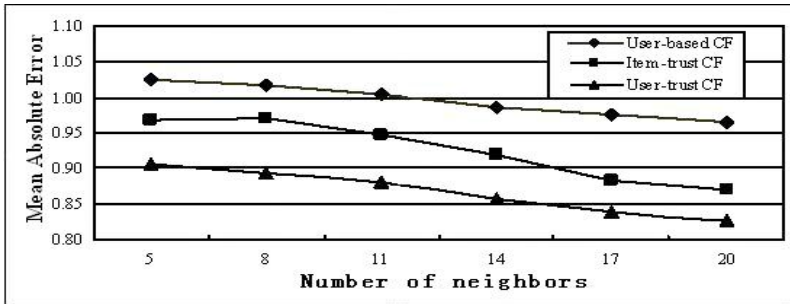


Fig. 2. Comparison of recommendation precision

Figure 2 shows the comparison of recommendation precision of three algorithms. The precision of item trust-based collaborative filtering algorithm and our user trust-based collaborative filtering algorithm is better than that of traditional user-based collaborative filtering algorithm, because they have considered the trust factor. The precision of our user trust-based collaborative filtering algorithm outperforms the item trust-based collaborative filtering algorithm. The performance is more remarkable with the number of neighbors increasing.

5.3 Robustness Analysis

In order to evaluate the robustness of the algorithm, we inserted some malicious ratings into the original data set. The rate of the malicious attack reached to 15%, namely the number of ratings achieved 252. But there were only 9.65% of the original users in the system reached the standard, under this condition there was only 91 ordinary user, a lower number compared with the malicious number. We selected 42 normal users as the test users whose ratings were over 320 so as to fully reflect the user’s hobby with the rating items and avoid the negative effects caused by lack of user hobby information. Figure 3 shows the comparison of attack resist capability of the three algorithms under different attack size. And the size of neighbor users takes 11 when calculating MAE.

In Figure 3, although the malicious users only occupy the recommendation user 5%, the influence on recommendation algorithms is significantly obvious.

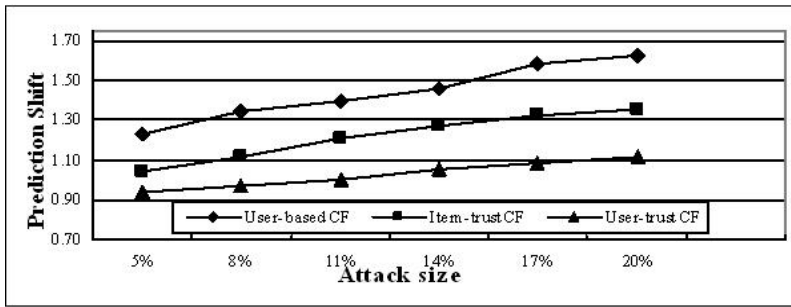


Fig. 3. Comparison of attack resist capability

Referencing figure 2 is not difficult to discover, the precision deviation separately expands from original (1.004, 0.947, 0.879) to (1.23, 1.04, 0.94). Because the traditional collaborative filtering algorithm generates recommendation only depends on the user ratings, the malicious users can provide a number of ratings which may become the similarity neighbors of the active user so as to have a great influence on its recommendation precision. Although item trust-based collaborative filtering algorithm and our user trust-based collaborative filtering algorithm also have some extent deviation, they introduce the trust pattern when they produce the recommendation neighbor, so they can effectively resist the influence of malicious noise. Thus the robustness of item trust-based and user trust-based collaborative filtering algorithm surpasses the traditional collaborative filtering algorithm in the face of malicious noise.

In Figure 3, when the attack size is smaller than 11%, the capability of attack resist for our user trust-based collaborative filtering algorithm outperforms the traditional collaborative filtering algorithm and item trust-based collaborative filtering algorithm. We notice that in general the user-trust approaches perform better than the item-trust approaches, especially the malicious attack size oversteps 11%. This is to be expected as the user-trust values provide a far more reliability of profile during recommendation and prediction. The reason of the phenomenon showed in Figure 3 is that the malicious users have a higher filled size with the rated items, and there would be more co-rating items with the target user, so that malicious users have more impact on the target users. But our user trust-based collaborative filtering algorithm is able to maintain a high precision in the face of malicious noise.

6 Conclusions

Traditional collaborative filtering recommendation algorithm is quite vulnerable in the face of malicious noise, which is a serious threat to the recommender systems that use collaborative filtering algorithm as an essential recommendation component. Recent research has showed that traditional collaborative recommendation algorithms can not ensure the precision of recommendations in the face of malicious noise.

In this paper, we have proposed an improved recommendation algorithm. It is an effective method to reduce the negative impact caused by malicious user ratings through the combination of user trust with the traditional user similarity. The user trust computation model and the corresponding user trust production algorithm are described. Combining user trust mechanism with traditional collaborative recommendation model can provide an effective way to defend against malicious noise. Experimental results have shown that the proposed algorithm is better in recommendation precision and resist-attack capability than the mentioned algorithm in this paper.

In the future we will improve our user trust-based collaborative recommendation algorithm and eliminate the vibration problem of the recommendation user trust value by introducing the detection mechanism of malicious user feature. This might be accomplished by filtering out bias recommendations or malicious neighbors in the recommendation algorithm.

References

1. Mobasher, B., Burke, R., Bhaumik, R., Sandvig, J.J.: Attacks and Remedies in Collaborative Recommendation. *IEEE Intelligent Systems*. J. 22(3), 56–63 (2007)
2. Williams, C.A., Mobasher, B., Burke, R.: Defending Recommender Systems: Detection of Profile Injection Attacks. *J. Service Oriented Computing and Applications* 1(3), 157–170 (2007)
3. Mehta, B., Nejdl, W.: Unsupervised strategies for shilling detection and robust collaborative filtering. *J. User Modeling and User Adapted Interaction* 19(1-2), 65–97 (2009)
4. Mehta, B., Hofmann, T.: A Survey of Attack-Resistant Collaborative Filtering Algorithms. *J. Data Engineering Bulletin* 31(2), 14–22 (2008)
5. Li, Y.-M., Kao, C.-P.: TREPPS:A Trust-based Recommender System for Peer Production Services. *J. Expert Systems with Applications* 36(2), 3263–3277 (2009)
6. Montaner, M., Lopez, B., de la Rosa, J.L.: Developing trust in recommender agents. In: 1st International Conference on Autonomous Agents, pp. 304–305. ACM Press, Bologna (2002)
7. Paolo, M., Paolo, A.: Trust-aware collaborative filtering for recommender systems. In: Meersman, R., Tari, Z. (eds.) OTM 2004. LNCS, vol. 3290, pp. 492–508. Springer, Heidelberg (2004)
8. John, O., Barry, S.: Trust in Recommender Systems. In: 10th International Conference on Intelligent User Interfaces, pp. 167–174. ACM Press, New York (2005)
9. John, O., Barry, S.: Is Trust Robust? An Analysis of Trust-Based Recommendation. In: 11th International Conference on Intelligent User Interfaces, pp. 101–108. ACM Press, New York (2006)
10. Deng, A., Zhu, Y., Shi, B.: A collaborative filtering recommendation algorithm based on item rating prediction. *J. Journal of Software* 14(9), 1621–1628 (2003) (in Chinese)
11. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* 4, 237–285 (1996)
12. O'Mahony, M.P., Hurley, N.J., Silvestre, G.C.M.: Detecting noise in recommender system databases. In: 11th International Conference on Intelligent User Interfaces, pp. 109–115. ACM Press, New York (2006)

Fingerprinting Attack on the Tor Anonymity System

Yi Shi and Kanta Matsuura

The University of Tokyo, Japan
{shiyi,kanta}@iis.u-tokyo.ac.jp

Abstract. We present a novel way to implement a fingerprinting attack against Onion Routing anonymity systems such as Tor. Our attack is a realistic threat in the sense that it can be mounted by a single controller of entrance routers and furthermore require very few resources. The conventional fingerprinting attack based on incoming traffic does not work straightforwardly against Tor due to its multiplex and quantized nature of traffic. By contrast, our novel attack can degrade Tor's anonymity by a metric based on both incoming and outgoing packets. In addition, our method keeps the fingerprinting attack's advantage of being realistic in terms of the few required resources. Regarding evaluation, the effectiveness of our method is discussed in a comprehensive manner: experimentally and theoretically. In order to enhance further studies and show the significance of our idea, we also discuss methods for defending against our attack and other applications of our idea.

Keywords: fingerprinting attack, anonymity system, Tor, onion routing.

1 Introduction

The internet brings us convenience, but also hurts our anonymity. With some tools, it is easy for an attacker to eavesdrop activities of other users. Individuals and organizations need anonymity on the Internet. People want to surf webpages, make online purchases, and send email without exposing their identities and activity patterns to others. Encryption solves some parts of this problem, but not everything. It can hide the communicating contents, but can do nothing with the packet headers, which reflects the identity of communication parties. Anonymity systems provides the foundation for users to share information over public networks without compromising their privacy.

A simple example: the websites nowadays keep profiles of users to provide a more suitable services. Large-scale B2C sites like Amazon will supply more suitable items for each user based on their surfing history and transaction records. If we bought some game software, then some games with the same platform will be recommended to us the next time. It makes seller provide better service and gives the buyer convenience, but it also hurts our privacy, since our transaction records could also be misused by the seller.

Anonymity systems could keep websites from profiling individual users. Anonymity systems could also be used for socially sensitive communication: forums or chat rooms for survivors of serious criminal cases, or people with specific illnesses. Journalists could use anonymity system to communicate with whistleblowers and dissidents safely. Corporations could use anonymity system as a safe way to conduct competitive analysis.

Moreover, big organizations such as embassies use anonymity systems to connect with their overseas headquarters. Law enforcement uses it for collecting evidence without alerting suspects. Non-governmental organizations usually use anonymity systems to connect to their website while they are abroad, without notifying everybody nearby what they are working with.

Modern anonymity systems was first introduced by Chaum as early as 1981 [4]. Many discussions followed that, both theoretical and practical, such as The Dining Cryptographers Problem [3], Babel [8], Crowds [13], Tarzan [7], MorphMix [14] etc. Among all of these anonymity systems, Tor [6] is one of the most widely used. It employs the Onion Routing scheme to anonymize the traffic, by using layered cryptography combined with widely distributed relay nodes in different administrative domains.

Although Tor is a well developed and still improving state-of-the-art anonymity system, it is vulnerable to traffic analysis attacks just like other low-latency anonymity system. Many researches concern about how to reveal the relationship between users who are using the anonymity system and how to degrade the anonymity. We believe that research on attacks against anonymity system will help us to understand the concept of anonymity more clearly, and help the anonymity system become more secure in the future.

The remaining parts are organized as follows: we will summarize the related works in Section 2, the attack plan in Section 3, and the experiments in Section 4. Countermeasures will be discussed in Section 5, and finally we will give the conclusion with some open questions in Section 6.

2 Related Works

After the early-day discussions about theoretical anonymity systems, many practical anonymity systems were presented in recent years as we discussed in Section 1. Practical anonymity systems trade off resistance against some kinds of adversaries with lower latency and better performance to provide usability to attract users. The mainstream of attacks toward anonymity systems are end-to-end confirmation attacks, which is mainly based on timing-based analysis techniques. It is widely researched and many papers exist, like [11,10]. It is a really powerful attack but also requires a strong assumption that the adversary controls both of the nodes adjacent to communication partners.

In low-latency anonymity system design, defense against timing attacks is merely considered as an desirable feature to achieve. Like in [6], Tor's designer claimed that known solutions seem to require either a prohibitive degree of traffic padding or an unacceptable degree of latency. Both will greatly decrease the usability of the anonymity system.

In 2008, Pries et al. presented a novel confirmation attack [12]. They do not use the classical way to analyze whether the two flows observed by two compromised nodes are identical, but just copy a packet from the flow by the entry node, and then choose the right time to replay it. The replayed packet could go through the whole path. And in the exit node, it will cause an integrity error, which is an unusual event. If the adversary could observe this event, he could assume that the two nodes are in the same path.

Although end-to-end confirmation attack dominates the attack research in this area, the disadvantage is also very clear - it is not practical enough. The assumption about control of both entry and exit node is unrealistic in the real world. Generally, we cannot be a global eavesdropper. The nodes in anonymity systems are usually distributed among the whole world. Even for big organization, such as governments or ISPs, it is also hard to control both ends of a path.

There are also other types of attacks towards anonymity systems. Fingerprinting attack was raised relatively early by Hintz in 2003 [9]. It uses different file sizes in a webpage to make a fingerprint, and try to use it to identify user's latter actions. It was first designed against SafeWeb. We will discuss it in detail later.

Chakravarty et al. presented another interesting idea in [2], 2008. In their threat model, the adversary needs not to be any point along the path. A completely separated attacker just observes all the bandwidths of nodes in the anonymity system using Linkwidth, a bandwidth-estimation technique. If several nodes's bandwidth changed by the same amount, then we could assume these nodes are in the same path. The disadvantage of this attack is that it assumed that the adversary occupies sufficient bandwidth and stands at a "vantage" point, which means that the bottleneck in the path connecting the adversary to the victim relay should always be the latter.

In [5][11], a well-known class of attacks called statistical disclosure treat the whole system as a black box. It correlates traffic that enters and exits it to discover some communication patterns. But the used model of anonymity system is somewhat simple and theoretical. The same as timing attacks, the threat model they used is out of anonymity system's consideration.

3 Fingerprinting Attack on Tor

In this section, we will first review the original fingerprinting attack, discuss the characteristic of Tor, and then raise our proposal of the fingerprinting attack on Tor.

3.1 The Original Fingerprinting Attack

Generally, when a user visits a typical webpage, it consists of many different files. First, the HTML file is downloaded from the site, after the links contained in the HTML file is analyzed by the browser and pictures included in the page, background music, flv movie, etc. would also be downloaded after that. If we surf the webpage at www.yahoo.co.jp, about 23 files would be retrieved from the server. Each of files has a specific file size.

In a typical browser, such as Microsoft Internet Explorer, each file would be downloaded via a separate TCP connection. So we can easily detect every TCP flows since they use different ports to transfer the files. Then, the attacker can determine the size of each file being returned to the client. All the attacker needs to do is just count the total size of the packets on each port.

This kind of attack can not only be applied to the plain flows, but also to the simple anonymity system like SafeWeb. Because common encryption methods do not try to obfuscate the transmitted data, for both performance and requirement reasons. If someone monitors the Safeweb user, the number and approximated file size could be determined. For example, the eavesdropper found that the user created 3 connections with the same target, each of the connections received respectively 1324 bytes, 582 bytes, and 32787 bytes. So each of these transfer sizes corresponds to a certain file directly, and the *fingerprint* of a webpage consists of the set of file sizes.

So the attacker could first try to build the fingerprint of the file sizes of webpages, then monitor the user. When the user is surfing a webpage, connections and related data could be detected by the attacker. Then the attacker just compare the connect data with a set of fingerprints, choose the closest one, then “guess” that the page is what user surfing now. The attack is low-cost, easy to apply, and really hurts the user’s anonymity.

3.2 The Characteristics of Tor

Tor is a low-latency, well developed anonymity system. It uses multi-hop encrypted connections to protect sender and/or receiver anonymity. Tor extends the former onion routing scheme by adding some features like integrity protection, congestion control, and location-hidden service. Tor can be used for both sender and receiver anonymity. Sender anonymity could help a user to use services without disclosing their identities. In Tor’s design, it employs two significant characteristics, which prevents the fingerprinting attack to some extent.

First, Tor employs quantized data cells, each data cell is fixed at 512 bytes. So it is obviously difficult for an attacker to detect the accurate size of files transferred by separated connection stream.

Second, Tor uses multiplexing to combine all the TCP streams into one connection. This is not for the safe aspect at first. The original Onion Routing creates a path for each TCP stream. But for the expensive communication cost, Tor decides to use multiplexing to reduce the expensive path-establish cost. And it also provides some resistance to the client against fingerprinting attacks, for the attacker cannot distinguish the connections between each other easily.

3.3 Threat Model of Fingerprinting Attack

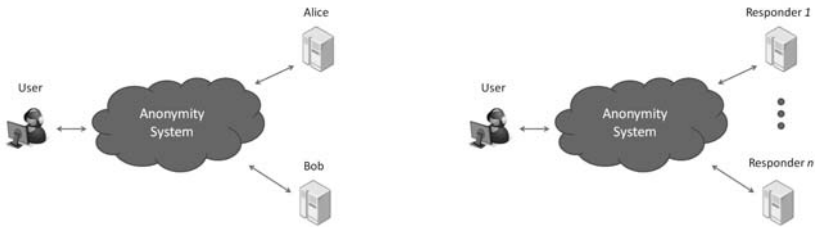
Although many attacks toward low-latency anonymity systems are successful in their assumed environment, Tor and other anonymity systems are considered to be secure in practical use. Many attacks involve a strong adversary, who could perform end-to-end confirmation or even global eavesdrop. And in practical world, it is obviously difficult to achieve this kind of requirement. Even for

big organizations to observe all the nodes distributed in the whole world is almost impossible. The advantage of fingerprinting attacks is the low resource requirement. The adversary only needs to occupy the entry point of the user. Compare to the end-to-end confirmation attacks, they just use the resources which much easier to satisfy, make it more possible to implement.

Our fingerprinting attack on Tor uses the same threat model with the fingerprinting attack by Hintz, the attacker is assumed to occupy the entry router of the user and observe all the data flows from the user. He wants to guess what webpage the user is surfing now. The design objective of Tor is attempting to defend against external observers who cannot observe both sides of a user's connections. So we think our threat model is appropriate against low-latency anonymity system.

Let us describe the model more formally, assume there is a user and two responders: Alice and Bob. An adversary can watch all the connections related to the user. First, the adversary could use the anonymity system to visit Alice and Bob for many times. Then the user visits either Alice or Bob using the anonymity system under the adversary's observation. Then the adversary would guess which responder the user connected to. We have some a priori probability, which models our suspicion about who is communicating with whom. More precisely, the a priori probability that the user is communicating with Alice is p and the a priori probability that user is communicating with Bob is $1 - p$. If we have no priori information, $p = 1/2$. See Figure 1(a).

Then, the model could also easily be extended to n responders, assume now there are n responders, from Responder 1 to Responder n . First, the adversary could use the anonymity system to visit any responder for many times. Then the user visits one responder using the anonymity system under the adversary's observation. Then the adversary would guess which responder the user connected to. We have some a priori probability, which models our suspicion about who is communicating with whom. More precisely, the a priori probability that user is communicating with Responder i is p and the a priori probability that the user is communicating with other responders are $1 - p$. If we have no priori information, $p = 1/n$. See Figure 1(b).



(a) Model with 2 responders

(b) Model with n responders

Fig. 1. Model in Fingerprinting Attack

3.4 Fingerprinting Attack on Tor

So we come to make our fingerprinting attack towards Tor. The biggest problem is that the only connection makes it hard for the adversary to distinguish each file size and the characteristic of the webpages becomes hard to define.

Generally, if we observe the traffic flow from/to the user, we will see a sequence of packets. If we use the outflow from user to separate the flow, we will see some interesting things. Some intervals may be very short, like 1 or 2 packets between two outflow packets. That means this interval transferred some small files or does some protocol transactions, etc. And some intervals may be relatively long, like 5 or 6 packets. This means a bigger file is being transferred. And after the TCP sliding window is fulfilled, the user send the acknowledge packet and continue the transfer process. If the network condition remains stable, this traffic pattern will not change much. So, for the webpages with different files and different loading process, we can distinguish them to some extent.

With a specific packet sequence, we could use the method described above to make a continuous intervals with the different number of packets. We call *all the inflow packets in a sequence, without any outflow packet placed in them*, an **interval**. We then define a vector $\mathbf{V} = (v_1, v_2, \dots, v_n)$, where v_i means “the number of intervals with i packets”. $n_{\mathbf{V}}$ means “the total number of intervals in \mathbf{V} ”. We build a fingerprint vector \mathbf{F} in advance. Let weight w defined as $n_{\mathbf{V}}/n_{\mathbf{F}}$ or $n_{\mathbf{F}}/n_{\mathbf{V}}$ which is smaller (equal when $n_{\mathbf{V}} = n_{\mathbf{F}}$) than 1. So we use this formula to calculate the similarity S :

$$S = \frac{\mathbf{V} \cdot \mathbf{F}}{\|\mathbf{V}\| \|\mathbf{F}\|} \cdot w \quad (1)$$

If we have several fingerprints, we could calculate observed \mathbf{V} with each \mathbf{F}_i to get several similarity S_i , then we could sort all the S_i and make the assumption the user is surfing the webpage with the \mathbf{F} correlated to the largest S_i .

In the ordinary fingerprinting attack, because a webpage is usually consists from about 20 to 30 files, and each file has its own unique file size. It means that the number of distinguishable webpages is very large. But in our work, the information we used is really limited due to the multiplexing. So if the number of fingerprint we use is too large, we may not have a very high detection rate. When the webpages the user may access are too many, after sorting the similarity S , we do not make the assumption only with the biggest S , but also using a threshold value θ instead. All fingerprints with calculated score larger than θ could be the possible page the user has seen. And we could make this as a set. If we could make sure the user is surfing the same page again and again (but we do not know which page he is watching), then we get other sets. Combine these sets and finally we could get the most possible answer.

3.5 The Choice of Fingerprints

So far we have discussed our threat model, the score formula and the method to recognize the page. But how can we choose a fingerprint?

Generally, any vector \mathbf{V} could be a fingerprint but the unique noises are also included in the fingerprint. An adversary may do the sampling work in advance and make a lot of vectors from one page. He wants to use them to achieve a higher detection rate from the data, so which one should he choose?

The fingerprint choosing method is also discussed in ordinary fingerprinting attack paper: the author claims that we should choose the smallest sizes sampled for each file. It is an intuitive idea that if we observed the same thing with the smallest size, then it must be with minimum noises. But in our opinion, for the adversary has almost same network condition as the user. The fingerprint should not only reflect the characteristic of webpage, but also the network condition of user.

We could assume the attacker access a webpage n times and recorded vectors as $\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_n$. We calculate the scores with each other by formula 1. Then we could get the scores S_{ij} calculated from \mathbf{V}_i and \mathbf{V}_j ($i = 1, 2, \dots, n-1, j > i$). So we could choose \mathbf{V}_i with the maximum S'_i as the fingerprint vector \mathbf{F} , which represents:

$$S'_i = \prod_{\substack{j \neq i \\ j}} S_{ij} \quad (2)$$

4 Evaluation

4.1 Environment and Data Collecting Method

We use Windump to capture the Tor packets (Version 0.2.0.34) on a PC with Intel Core2 Duo 1.86G, 4G RAM, Vista Business. We shall run the windump to observe the port 9001 on the host machine. Then we use firefox which installed TorButton to surf the webpage. After a webpage is fully loaded, we stop capturing the packets. We use Wireshark to open the pcap file, filter the obvious noise manually. More precisely, in a short period, all the connections raised from Tor are going through the same path. So most of the packets will obviously have the same destination address (Actually, this address refers to the first node in the path). And some packets with other destination addresses refer to other control packets used in Tor, like establishing new paths. After this process, a data is recorded. We also wrote some programs to analysis the captured data to make the calculation.

4.2 Data Analysis

First, we shall use Alexa Ranking - Top Sites in Japan¹ to see how our method works in a practical environment. In Figure 2, we use n to represent the top n sites' mainpages we used to implement the experiment. We choose the top 20 sites to implement the experiments.

In the experiment, we choose top $n = 5, 10, 15, 20$ sites, and built fingerprint of the site. Then we surfed webpages and recorded the user activity vector, compared with the fingerprint, and guessed which website user is surfing. The success rate represents in the Figure 2.

¹ <http://www.alexa.com/topsites/countries/JP>

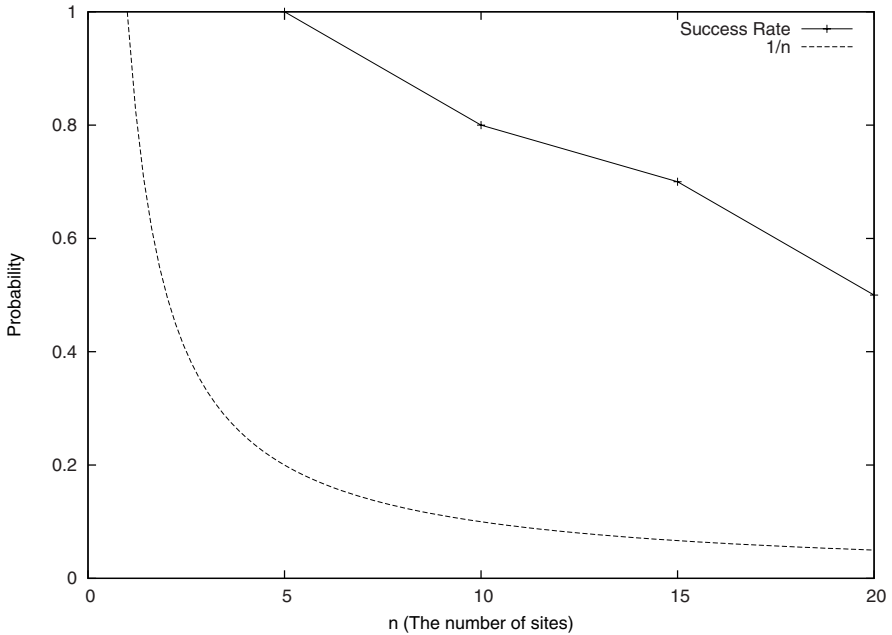


Fig. 2. Success Rate in Different n

From Figure 2, we could see that: the success rate is relatively high when n is small. With n increases, the success rate decreases significantly. There are several reasons for this: First, the information we used is limited, the fingerprint of the webpage is not so unique. So obviously the success rate decreases when n increases. Second, some pages are not suited for fingerprinting, like youtube², amazon³. The items on these sites would change from time to time, which hurts the consistency of fingerprinting. Other sites like Yahoo⁴ will have ads change frequently, too. But compared with other parts of the page, the ratio of ads is not so large and we could just treat them as noise. Third, the pages we have chosen are all homepages, with the similar design, it increases the difficulty of distinguishing. Fourth, the noise in practical network affects the result a lot, and that's why we need to implement our method instead of just making the simulation. Last, there are some sites hard to see the difference but still be counted as different ones, like Google⁵ and Google Japan⁶. This problem also exists in the distinguishing between the original page and phishing page. We will discuss the success rate in more formal way in the following section.

² <http://www.youtube.com/>

³ <http://www.amazon.co.jp/>

⁴ <http://www.yahoo.co.jp/>

⁵ <http://www.google.com/>

⁶ <http://www.google.co.jp/>

4.3 Theoretical Discussion

In this part, we will discuss the effectiveness of this attack in theory. We will discuss two topics: The factors that related to success rate and make an estimate of how many webpages (or webpage groups) could be distinguished without too high error rate.

First, Let us discuss with the success rate. We will use the method in [10] to show the attack success probability formally: We use $V \sim F$, to indicate that the attacker’s test says that vector V and fingerprint F are from the same site. And we use $V = F$ to indicate that the event that vector V and fingerprint F are from the same site. We have the *false positive rate*, $Pr_{fp} = Pr(V \sim F|V \neq F)$, and *false negative rate*, $Pr_{fn} = Pr(V \approx F|V = F)$, are both known. We can therefore obtain:

$$\begin{aligned} Pr(V \sim F) &= Pr(V \sim F|V = F)Pr(V = F) + Pr(V \sim F|V \neq F)Pr(V \neq F) \\ &= (1 - Pr_{fn})Pr(V = F) + Pr_{fp}(1 - Pr(V = F)) \\ &= (1 - Pr_{fn} - Pr_{fp})Pr(V = F) + Pr_{fp} \end{aligned}$$

Which leads us to obtain:

$$\begin{aligned} Pr(V = F|V \sim F) &= \frac{Pr(V = F \wedge V \sim F)}{Pr(V \sim F)} \\ &= \frac{Pr(V \sim F|V = F)Pr(V = F)}{Pr(V \sim F)} \\ &= \frac{(1 - Pr_{fn})Pr(V = F)}{(1 - Pr_{fn} - Pr_{fp})Pr(V = F) + Pr_{fp}} \end{aligned} \tag{3}$$

Suppose $Pr(V = F) = 1/n$, e.g., we are observe n sites and the adversary has no additional information about which site the user is likely surfing. Then, the success probability depends on Pr_{fp} and Pr_{fn} .

In the simplest case, we first assume the false positive rate and false negative rate are constant. Then, with $Pr_{fn} = Pr_{fp} = 0.1$ and $n = 10$, which means the user could surf 10 webpages and we’ve made all the fingerprints of them, we could get $Pr(V = F|V \sim F) = (0.9 \cdot 0.1)/(0.8 \cdot 0.1 + 0.1) = 50\%$. And if we improve Pr_{fn} and Pr_{fp} to 0.01, then with 10 webpages, the success probability is about 91.7%. As n rises to 100 webpages, this probability also falls to only 50%. With $n = 1000$, it is less than 10%.

But as we see in the evaluation above, the Pr_{fn} and Pr_{fp} rises with n . So, we will describe the false positive rate and the false negative rate as a function of n . We also use the assumption $Pr(V = F) = 1/n$ discussed above. Then the Equation 3 would be:

$$\begin{aligned} Pr_{Success} &= \frac{(1 - F_{fn}(n))/n}{((1 - F_{fn}(n) - F_{fp}(n))/n) + F_{fp}(n)} \\ &= \frac{1 - F_{fn}(n)}{1 - F_{fn}(n) - F_{fp}(n) + F_{fp}(n) \cdot n} \end{aligned} \tag{4}$$

Actually, it is almost impossible to make a reasonable function to reflect the relationship between n and the error rate, for it is affected greatly by the sites we have chosen. But we could assume Pr_{fn} and Pr_{fp} have a linear relationship with the increase of n , then from the Equation 4, we could see the numerator falls with n , and the denominator increases even faster, which will leads the success probability decreasing even faster.

The equations we listed above tell us if we want to increase the success rate, there are several points: First, to improve the accuracy, that is, decrease the false positive and false negative rate. Second, make the webpages we need to guess as few as possible, what means make the n lower. What's more, we assume the adversary knows nothing in advance. So the $Pr(V = F)$ equals $1/n$. But if in some situation, $Pr(V = F)$ is greater than $1/n$, which means the adversary gets some additional information from other ways, the success rate itself will also be raised.

Then, we shall come to how many webpages we could distinguish without high error rate, if not choose the webpages randomly but we could choose by ourselves.

Notice that the similarity S consists of two components, the relative interval ratio and the vector's dot product. First, we take a look at the relative interval ratio. We have implemented an experiment to get that the mainpage of Yahoo Japan have an average interval of 159.2105, with the standard deviation of 14.8495. Figure 3 shows the distribution of intervals of Yahoo Japan.

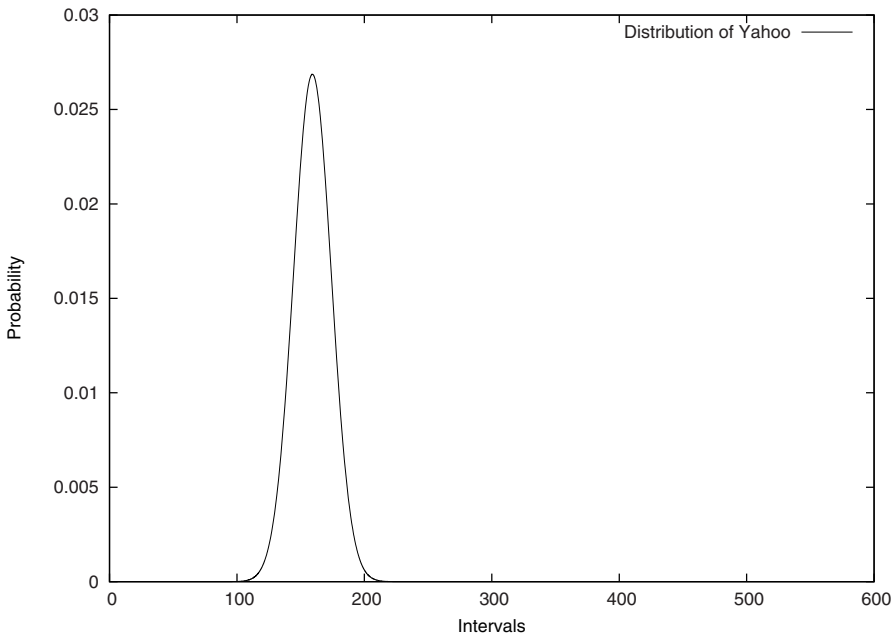


Fig. 3. Distribution of Intervals of Yahoo Japan

From our observation, the intervals of webpages often fall in the range from 50 to 600. We can choose the page freely here, webpages with more than 1000 intervals are not so rare in practical. But here we just want to make an theoretical estimate; we will choose the range of interval up to 600.

As our experiment about Yahoo Japan, the standard deviation is approximately 10% of the intervals, that means, with about $\pm 20\%$ gap between two sites, there is about 95% chance the vector could be recognized correctly. Roughly speaking, there are $\log_{1.4}(600/50) + 1 \approx 8.38$ slots for us to choose webpages with high detection rate.

Then we come to the dot product of vectors. In our implementation, the vector is limited to 5-dimension. Because intervals with more than 5 packets are so rare, intervals with more than 5 packets would be treated as one with just 5 packets.

Theoretically speaking, if we use 20% gap as we do in the discussion about interval, then there are a lot of available slots for us to choose, considering we have 5-dimension to do the permutation. But actually, in typical situation, the intervals with 1 or 2 packets dominated in the total dimensions, for there are a lot of transactions to be done (Also, in some extreme condition, such as file transferring, we could expect to observe a lot of long intervals). By our observation, in the situation with similar intervals, there are about 3 or 4 significantly different results. Combine this with the result about interval, we have approximately 20 to 40 available slots for choosing webpages to be recognized.

It is hard to improve this result, unless we could find some way to significantly reduce the noise. But it is expected to be improved by following research. For example, our method does not employ with time/latency, if we could employ it into the calculation of score, the number of pages we could detect maybe greatly increased.

5 Countermeasures and Discussions

In this section, we will discuss some countermeasures to our attack and some countermeasures which is believed to be effective toward fingerprinting attack and its possible extensions. And there are also some open questions waiting to be further researched.

5.1 Change the Fixed Cell Size

It is believed a longer Tor cell size will make it harder to attack, e.g. Increase the Tor's cell size from 512 bytes to 1024 bytes. But unfortunately, in our attack scheme, it will have little impact. Tor's fixed cell size gives the system some advantage in traffic analysis theoretically. But the protocol it uses is still built on TCP. So no matter what the cell size is, it could still be wrapped by TCP packet and be divided into 1500 bytes a packet in ethernet. If there exists a scheme to analysis Tor's cell from TCP packets, this defense method could have some results, but not in our proposal.

5.2 Make Odd Requests

Odd requests refers to some surfing actions which is unusual. For example, always surfing several pages meanwhile, restricting the scripts or pictures downloading, etc. If there is a page with vector V_1 and another with vector V_2 , then when we view these two pages at same time, the adversary could get a V_3 equals $V_1 + V_2$, and V_3 has no difference with the vector V'_3 which has the same elements as V_3 , although it may be observed from one single page. Other odd requests like the restriction on downloading some specific files. Like the combination of two pages, it is also difficult for an adversary to match the characteristic from the fingerprint vector. Although this kind of defensive method seems to be so effective against fingerprinting attack, it depends on the user's action. But we cannot make the system's security depends how users use this system. It is dangerous to assume the users have the knowledge in security and will work in a secure way. Moreover, it is not hard to develop some kind of explorer plug-in to achieve this objective. Like TorButton, if we activate this plug-in, it will randomly disable some kinds of files in the current webpage, maybe forbid running script or download pictures. It will help us in the anonymity, but we do not think users would really accept some plug-ins like this.

5.3 Run Own Entry Node

Entry nodes are also called "guard nodes". And people believed that they could guard your traffic from malicious nodes. First, it is not so useful to run a node by oneself when the adversary occupies the entry router. Especially the time when they are allocated in the same ethernet. Second, to run an own entry node and achieve the requirement of anonymity is very costly. That means, to make an adversary unable to distinguish the flows from a user. The own node may accept many connections from other users, which may hurt the usability of company's network and unacceptable. But running a node with only permitted user also makes this node meaningless. How to make the balance could be a question to network administrators.

5.4 Defensive Dropping

Defensive Dropping is a defensive method against timing attacks introduced by Levine et al. [10]. It employs the mechanism of dummy packets. The communication initiator constructs some of the dummy packets. These dummy packets are transferred on the path as normal packets. But to each packet, there is a probability P_{drop} to be dropped in each node rather than passing it on to the next node. If the number of dummy packets is randomly placed with a sufficiently large frequency, the correlation between every visiting will be greatly reduced. As we see in this theoretical discussion part, the increasing in the false positive rate and false negative rate will greatly reflected in the situation where we need to recognize object from a lot of webpages.

Although it is an effective way to defend against not only end-to-end attacks but also fingerprinting attack, we must notice that it is a really expensive defense

mechanism, especially in low-latency anonymity system. If the number of the dummy packets is relatively small, then these dummy packets are no more than normal background traffics. But with many dummy packets, it is unacceptable for consuming so many resources. What's more, use more dummy packets in sensitive connection is also not a good idea for it gives the adversary a clear sign to notice the sensitive data transfer. So how to determine the sufficient number of packets will leave to be an open question for further research.

5.5 Other Applicable Situations

Although this attack is mainly designed towards Tor, it could also be applied in other situations.

First, it is not hard to see every anonymity systems with multiplexing or quantized cells could be attacked by our proposal. And even without multiplexing or quantizing, our proposal also works. You can treat all the connections as if they were one. But it would be ineffective for we discard some useful information by this process.

Second, this kind of attack can not only be applied attacking information regarding webpage surfing, but also other forms of network activities. For example, in instant chatting, there should be differences between one who talks quickly but every sentence is short and another merely talks but using long paragraphs. This kind of differences could be reflected in their traffic flows, although the significance may not be high enough to be detected.

What's more, our scheme do not only apply to the entry point of the path, but also the exit point. Imagining that if you are a curious server administrator who is running a system which accepts both anonymous and non-anonymous visits from anonymity systems. You could record the patterns when users visiting your sites in non-anonymous mode. And someday, for some purpose, a user visits your sites anonymously. Then you could use this scheme to guess which user it is. Just by comparing the historical patterns and the flows you observed.

We just simply described some other possible situations for the application of our proposal. Theoretically, for any kinds of activities with stable traffic patterns, our proposal could be a potential threat. Although the effectiveness of our proposal still needs to be improved by further research.

6 Conclusion

In this paper, we have discussed a novel fingerprint attack against the Tor anonymity system. Our scheme works by analyzing users' traffic flows in the anonymity system. We use outflow packets to divide a flow into several intervals, turn a flow into a vector, and give a formula to calculate the similarity of two vectors in this scheme. It can easily be implemented by network administrators, governments, or ISPs. The experimental results showed our scheme to be very effective. The user's anonymity is really degraded by this simple and practical attack. As we have discussed, this effectiveness has a potential of being improved even more. Also, we have given a theoretical reasonable estimate of the

effectiveness, showed the simple model of fingerprinting attacks on anonymity systems. Moreover, the result showed the need for the use of dummy traffic in the low-latency anonymity systems, but how to evaluate it is still left as an open question. As future work, we may improve the attack method, especially the scalability. We could also try to present some new threat models.

References

1. Back, A., Möller, U., Stiglic, A.: Traffic analysis attacks and trade-offs in anonymity providing systems. In: Moskowitz, I.S. (ed.) *IH 2001*. LNCS, vol. 2137, pp. 245–257. Springer, Heidelberg (2001)
2. Chakravarty, S., Stavrou, A., Keromytis, A.D.: Identifying proxy nodes in a tor anonymization circuit. In: *Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, Washington, DC, USA, pp. 633–639. IEEE Computer Society, Los Alamitos (2008)
3. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology* 1, 65–75 (1988)
4. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24(2), 84–90 (1981)
5. Danezis, G.: Statistical disclosure attacks: Traffic confirmation in open environments. In: *Security and Privacy in the Age of Uncertainty, International Conference on Information Security (SEC 2003)*, Athens, Greece, May 26–28, pp. 421–426. Kluwer, Dordrecht (2003)
6. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: *Proceedings of the 13th USENIX Security Symposium* (August 2004)
7. Freedman, M.J., Morris, R.: Tarzan: a peer-to-peer anonymizing network layer. In: *Proceedings of the 9th ACM conference on Computer and Communications Security*, pp. 193–206. ACM, New York (2002)
8. Gülcü, C., Tsudik, G.: Mixing E-mail with Babel. In: *Proceedings of the Network and Distributed Security Symposium - NDSS 1996*, pp. 2–16. IEEE, Los Alamitos (1996)
9. Hintz, A.: Fingerprinting websites using traffic analysis. In: Dingledine, R., Syverson, P.F. (eds.) *PET 2002*. LNCS, vol. 2482, pp. 171–178. Springer, Heidelberg (2003)
10. Levine, B.N., Reiter, M.K., Wang, C., Wright, M.K.: Timing attacks in low-latency mix-based systems. In: Juels, A. (ed.) *FC 2004*. LNCS, vol. 3110, pp. 251–265. Springer, Heidelberg (2004)
11. Mathewson, N., Dingledine, R.: Practical traffic analysis: Extending and resisting statistical disclosure. In: Martin, D., Serjantov, A. (eds.) *PET 2004*. LNCS, vol. 3424, pp. 17–34. Springer, Heidelberg (2005)
12. Pries, R., Yu, W., Fu, X., Zhao, W.: A new replay attack against anonymous communication networks. In: *Proceedings of the IEEE International Conference on Communications, ICC 2008*, May 2008, pp. 1578–1582 (2008)
13. Reiter, M., Rubin, A.: Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security* 1(1) (June 1998)
14. Rennhard, M., Plattner, B.: Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In: *Proceedings of the Workshop on Privacy in the Electronic Society*, Washington, DC, USA (November 2002)

Proactive Verifiable Linear Integer Secret Sharing Scheme

Chuangui Ma and Xiaofei Ding

Zhengzhou Information Science and Technology Institute,
Zhengzhou, 450002, China
chuanguima@yahoo.com

Abstract. The commonly used technique for cheating detection in verifiable secret sharing (VSS) require public key systems. Based on linear integer secret sharing (LISS) scheme, this paper presents a private verifiable protocol over arbitrary access structure without public key systems, which can avoid cheating both from participants and dealers. For further consideration of share refreshing and renewal, this paper shows the proactive property of our scheme with new method. Furthermore, this paper applies combinatorial structure into the proactive scheme to reduce the time of the computation.

Keywords: LISS; refresh; renewal; proactive property; combinatorial structure.

1 Introduction

One important topic in cryptography is how to securely share a secret among a group of people. In a secret sharing scheme, a *dealer* distributes the secret to a number of shareholders, such that only qualified sets can reconstruct the secret, while other subsets have no information about it. It is a fundamental building block for many cryptographic protocols and is often used in the general composition of secure multiparty computations. The collection of consisting of qualified sets is called the access structure.

Blackley, G.R. [1] and Shamir, A. [2] independently introduce the first Secret Sharing in 1979, which store critical information such that we get at the same time protection of privacy and security against losing the information. Later, secret sharing has proved extremely useful, not just as a passive storage mechanism, but also as a tool in interactive protocols. So it's a important and useful tool to make a good secret sharing scheme.

Linear integer secret sharing (LISS) was introduced by Damgard, I. and Thorbek, R. [3]. In LISS scheme, the secret was an integer chosen from a (publicly known) interval, and each share was computed as an integer linear combination of the secret and some random numbers chosen by the dealer. Reconstruction of the secret was also by computing a linear combination with integer coefficients of the shares in a qualified set.

Based on the concept of an integer span program (ISP) introduced by Cramer et al. shown that any ISP could be used to build a private secure LISS scheme. The details could be find in [4,5]. Private security and perfect security were different concepts in some cases and they were shown in [6,7] that perfect secret sharing and private computation over countably infinite domains (like the integers) were not possible. However, this didn't rule out schemes of this type since the secrets were chosen from a publicly known interval, and protocols were proved to be statistical security rather than perfect privacy.

Another security aspect in secret sharing schemes is cheating prevention. There are two ways to do this. One method uses longer shares as in [21]. The other requires extra information to verify the shares of the shareholders. There are many papers investigate such schemes [10,11]. The verifiable secret sharing schemes depended on some cryptographic assumptions. In Pedersen's scheme, the privacy of the secret was unconditionally secure, but the correctness of the shares based on a computational assumption.

The third security consideration is share refreshing and renewal. In some occasions, a secret value (for example a cryptographic master keys, data files and legal documents) should to be stored for a long time. In this case, an adversary attacked the locations one by one and eventually got the secret or destroyed it. To resist such attack, proactive secret sharing schemes were proposed. Proactive security for secret sharing was first suggested by Ostrovsky, R. and Yung, M. [8]. Their paper presented a proactive polynomial secret sharing scheme. Proactive security refers to security and availability in the presence of a mobile adversary. Herzberg, A. et al. [9] specialized this notion to robust secret sharing schemes and gave a efficient proactive secret sharing scheme.

Our contribution are three fold: the first is verifiable, the second is proactive and the third optimize. In this paper a new proactive secret sharing scheme is proposed. Shares are periodically renewed without changing the secret. Every participant is able to verify the share which he receives and those other participants show. This scheme can prevent adversaries from getting the secret or sharing and the participants cheating from each other efficiently.

Moreover, we introduce some combinatorial structures [12] in the scheme so that the scheme will be more efficient. With uses of combinatorial structures, we can obtain a predetermined arrangement of the servers which permits the possibility of reducing the computation of the scheme. Our scheme is more efficient in the situation when the number of the possible corrupted servers are much smaller as compared to the total number of the servers in the system.

The remainder of this paper is organized as follows. In Section 2 we give some preliminaries and recall the LISS scheme. Section 3 describes our Verifiable LISS scheme. In Section 4 we provide a Proactive secret sharing. Section 5 gives an security analysis. Section 6 introduces Verifiable LISS scheme with combinatorial structure and analyzes the efficiency of our scheme briefly. Finally, conclusions is presented in Section 7.

2 Secret Sharing Scheme

2.1 Preliminaries

First we describe the definition of the access structures from [5].

Definition 1. [5] *A monotone access structure on $\{1, \dots, n\}$ is a non-empty collection Γ of sets $A \subseteq \{1, \dots, n\}$ such that $\emptyset \notin \Gamma$ and such that for all $A \in \Gamma$ and for all sets B with $A \subseteq B \subseteq \{1, \dots, n\}$, it holds that $B \in \Gamma$.*

In the following we define the notion of an Integer Span Program (ISP, introduced in [5]) and show how any ISP can be used to build a correct and private LISS scheme in [13].

Definition 2. [5] *$\mathcal{M} = (M, \psi, \varepsilon)$ is called an Integer Span Program (ISP), if $M \in Z^{d \times e}$ and the d rows of M are labelled by a surjective function $\psi : \{1, \dots, d\} \rightarrow \{1, \dots, n\}$. Finally, $\varepsilon = (1, 0, \dots, 0)^T \in Z^e$ is called the target vector. We define size $(\mathcal{M}) = d$, where d is the number of rows of M .*

Definition 3. [13] *Let Γ be a monotone access structure and let $\mathcal{M} = (M, \psi, \varepsilon)$ be a integer span program. Then \mathcal{M} is an ISP for Γ , if for all $A \subseteq \{1, \dots, n\}$ the following holds.*

1. *If $A \in \Gamma$, then there is a vector $\lambda \in Z^d$ such that $M_A^T \lambda = \varepsilon$.*
2. *If $A \notin \Gamma$, then there exists $\kappa = (\kappa_1, \dots, \kappa_e)^T \in Z^e$ such that $M_A \kappa = 0 \in Z^d$ with $\kappa_1 = 1$, which is called the sweeping vector for A .*

Then we review the definition of Verifiable Secret Sharing protocol in [14].

Definition 4. [14] *A player execute protocol honestly is called a good player.*

Definition 5. [14] *The protocol π is an private Verifiable Secret Sharing protocol if the following properties are hold:*

1. *If a good player P_i outputs $ver_i = 0$ at the end of Share then every good player outputs $ver_i = 0$;*
2. *If the dealer is good, then $ver_i = 1$ for every good P_i ;*
3. *If at least $n - b$ players P_i output $ver_i = 1$ at the end of Share, then there exists an $s' \in S$ such that the event that all good P_i outputs s' at the end of Reconstruct is fixed at the end of Share and $s' = s$ if the dealer is good ;*
4. *For any two secrets s, s' , any forbidden set A of shareholders. Set vector $r_i = M_i \rho$ for $i = 1, \dots, t - 1$, $r_i \in [0 \dots 2^l]$ we can find symmetrical matrix ρ , set $r_i = M_i \rho'$, where r, r' are statistically indistinguishable. More precisely, the statistical distance between the two distribution is negligible in k .*

2.2 Linear Integer Secret Sharing

The materials of this subsection come from [13].

Let $P = \{1, \dots, n\}$ denote the n shareholders (or players) and D as the dealer. The dealer D wants to share a secret s from the publically known interval

$[0 \dots 2^d]$ to the shareholders P over Γ . such that every set of shareholders $A \in \Gamma$ can reconstruct s , but such that a set of shareholders $A \notin \Gamma$ gets no or little information on s .

There is an adversary which can corrupt at most b servers at most during any time period. Corrupting a server means learning the secret information of the server, modifying its data, sending out wrong message, changing the intended behavior of the server, disconnecting it, and so on. Since the server can be rebooted, the adversary is a mobile one.

A secret value $s \in GF(q)$ will be shared by the servers through the scheme. The value of s needs to be maintained for a long period of time. The life time is divided into time periods which are determined by the global clock. At the beginning of each time period the servers engage in an interactive update protocol. The update protocol will not reveal the value of s . At the end of the period the servers hold new shares of s . The mobile adversary who corrupts b servers in a time period cannot get any information about the secret value s . The system can reproduce s in the presence of the mobile adversary at any time.

Share

We use a distribution matrix $M \in Z^{d \times e}$ and a distribution vector $\rho = (s, \rho_2, \dots, \rho_e)^T$, where s is the secret, and the ρ_i^s are uniformly random chosen integers in $[0 \dots 2^{l_0 + \kappa}]$ for $2 \leq i \leq e$, where κ is a security parameter and l_0 is a constant that is part of the description of the scheme. The dealer D calculates shares by

$$M \cdot \rho = (s_1, \dots, s_d)^T$$

where we denote each s_i as a share unit for $1 \leq i \leq d$. Let $\psi : \{1, \dots, d\} \rightarrow P$ be a surjective function. The i 'th share unit is then given to the i 'th shareholder, we say that $\psi(i)$ owns the i 'th row in M . If $A \subseteq P$ is a set of shareholders, then M_A denotes the restriction of M to rows jointly owned by A . We denote d_A as the number of rows in M_A . Similarly, for $s \in Z^d$ let $s_A \in Z^{d_A}$ denote the restriction of s to the coordinates jointly owned by A .

Reconstruct

For a qualified set A , there is $\lambda_A \in Z^{d_A}$ which gives $M_A \lambda_A = 0 \in Z^{d_A}$,

$$s_A^T \lambda_A = (M_A \rho)^T \cdot \lambda_A = \rho^T \cdot (M_A^T \cdot \lambda_A) = \rho^T \cdot \varepsilon = s$$

From [3] we know that the LISS scheme is correct and private. But secret sharing have two problems: one is the security of initialization, although we think Dealer as a trusty center in many instances, there are many unpredictable factors practicality, which is difficult to guarantee transmission correctly in network and malicious attack dealer. Another aspect is efficiency of share units, i.e. how to ensure the correctness of share units when transmitted from each other. The next section will show the method to solve these problems.

3 Verifiable Secret Sharing

To prevent cheating behaviors among secret sharing and recovering. we present a verifiable LISS scheme with a different method from traditional verifiable schemes.

Before reconstructing secret, combiner reconstructor first validate share units from other participators. The following is the detail protocol.

1. *Dealer* random construct a symmetrical matrix $\rho = (\rho_0, \dots, \rho_{e-1}) = (\rho_{ij})_{e \times e}$, where $\rho_{00} = s$, M is matrix of ISP. Let $M\rho_{00} = (s_{01}, \dots, s_{0d})$ be share units and $M\rho_i$, for $i = 1, \dots, e - 1$ as public verifiable vectors.

2. P_i sends $s_{\psi(i)} = M_{\psi(i)}\rho_0$ to P_j ,

3. After receiving $M_{\psi(i)}\rho_0$, P_j checks whether $M_{\psi(i)}\rho M_{\psi(j)} = M_{\psi(j)}\rho M_{\psi(i)}$, if P_j finds that $M_{\psi(i)}\rho M_{\psi(j)} \neq M_{\psi(j)}\rho M_{\psi(i)}$, P_j broadcasts (i, j) .

4. Each P_i computes the maximum subset $G \subseteq \{1, \dots, n\}$ such that any ordered pair $(i, j) \in G \times G$ is not broadcasted, If $|G| \geq n - b$, then P_i outputs $ver_i = 1$; otherwise, P_i outputs $ver_i = 0$, and requires *Dealer* repeat share secret.

It is obvious that every good player computes the same subset G in the end of *share*. The reconstruct phase that is the same as stated above.

4 Proactive Secret Sharing

It is dangerous for long live periodic secret. The most efficient method is processing. This section introduces share renewal in period to protect this kind of secret. So in this situation, we can divide life time into time periods: mark the length of each time period as t , time of share renewal at beginning and end phase, and keep secret changeless after share renewal.

4.1 Share Renewal Protocol

Each P_i for $1 \leq i \leq n$ random choose vector $\rho_i = (0, \rho_{i2}, \dots, \rho_{ie})$, calculates shares by $M\rho_i = (s_{i1}, \dots, s_{id})$, where s_{ij} is given to $p_{\psi(j)}$, i.e. $p_{\psi(j)}$ receive (s_{1j}, \dots, s_{nj}) , renewal share as $s_j^t = s_j^{t-1} + s_{1j} + \dots + s_{nj}$.

Reconstruct secret: For a qualified subset A we have that

$$(s_A^t)^T \lambda_A = (s_1^t, \dots, s_{d_A}^t)^T \lambda_A = (s_1^{t-1} + s_{11} + \dots + s_{n1}, \dots, s_{d_A}^{t-1} + s_{1d_A} + \dots + s_{nd_A})^T \lambda_A = (s_1^{t-1}, \dots, s_{d_A}^{t-1}) \lambda_A + (s_{11} + \dots + s_{n1}, \dots, s_{1d_A} + \dots + s_{nd_A}) \lambda_A = s$$

4.2 Detection of Corrupted Shares

In the proactive secret sharing system, users must be able to ensure that shares of other users have not been corrupted or lost, and be able to restore the correct shares if necessary. Otherwise, an adversary could cause the loss of the secret by destroying shares. This subsection presents a mechanism for detection of corrupted shares.

The idea is to save some fingerprint for each share that is common to all the shareholders, so that periodically, shareholders can compare shares (using secure broadcast). In order to implement the distributed verifiability of shares, a basic feature is added to the previous protocol. In each time period, each user stores the encryptions of all the shares he/she received from the other users.

- Definition 6.**
1. P_i sends $M_{\psi(i)}\rho$ to $P_j, j = 1, \dots, n, j \neq i$;
 2. P_j checks whether $M_{\psi(i)}\rho M_{\psi(j)} = M_{\psi(j)}\rho M_{\psi(i)}$, then P_j broadcasts an accusation list j which contains those i such that $M_{\psi(i)}\rho M_{\psi(j)} \neq M_{\psi(j)}\rho M_{\psi(i)}$ or $M_{\psi(i)}\rho$ was not received.
 3. Each good player updates the list \mathcal{L} so that it contains those i accused by at least $b + 1$ players of the system.

4.3 Recovery of Lost/Corrupted Shares

This is a fundamental phase in the proactive scheme, because without it, this scheme would not be secure against adversaries who disable some users from performing the required protocol.

After running *detection*, the system will recover the shares for all players P_l , where $l \in \mathcal{L}$. The recovery protocol is as follows.

1. For each $l \in \mathcal{L}$, every good players P_i sends $M_{\psi(i)}\rho_0^k$ to P_l ;
2. Upon receiving the data, P_l computes $M_{\psi(l)}\rho^k M_{\psi(i)} = M_{\psi(i)}\rho^k M_{\psi(l)}$, P_l sets $M_{\psi(l)}\rho_0^k + M_{\psi(l)}\rho_0$ as its shares.

5 Security Analysis

The private and verifiable of our scheme can be illuminated from the follow theorem.

Theorem 1. *The above LISS scheme is private and Verifiable.*

Proof. We prove that the above scheme satisfies the conditions of the definition 5 as follows:

1. If a good player P_i outputs $ver_i = 0$, then the size of the maximum subset G is at most $n - b - 1$. Thus every good player will output 0;
2. If the dealer is good, then good player receives $M_{\psi(i)}\rho^0$. Since ρ is a symmetric matrix, $M_{\psi(i)}\rho M_{\psi(j)} = M_{\psi(j)}\rho M_{\psi(i)}$ for all good players P_j , Thus all good players are in the subset G . Therefore $ver_i = 1$ for every good P_i ;
3. Suppose at least $n - b$ players output "1" at the end of the Share. Then there is a subset G of size $n - b$ such that no one in the subset complained the others. Since we assume that there at most b bad players, there are at least $n - 2b$ good players in G , in which who all of them have consistent shares. Thus there is a qualified set $A, \lambda_A \in Z^{d_A}$, st:

$$s' = s_A^T \lambda_A = (M_A \rho)^T \cdot \lambda_A = \rho^T \cdot (M_A^T \cdot \lambda_A) = \rho^T \cdot \varepsilon = s$$

4. For arbitrary s We have chose $\rho = (\rho_0, \rho_1, \dots, \rho_{e-1})$, with $\rho_{00} = s$, $\rho_{ij} \in [0 \dots 2^{l_0+\kappa}]$ as uniformly random numbers, and secret $s \in [0 \dots 2^l]$.

Let $s' \in [0 \dots 2^l]$ be arbitrary, We first observe that $s_A = M_A \rho$ is the subset of shares that belongs to A . If A is a forbidden set, there exists a sweeping vector κ , such that $M_A \kappa = 0 \in Z^{d_A}$.

Define $\rho' = (\rho_0 + (s' - s)\kappa, \rho_1, \dots, \rho_{e-1})$, then we have $M\rho = M\rho'$, that is the shareholders in A see the same shares, but the secret s' was shared instead of s . Define ρ is good if $\rho' = (\rho_0 + (s' - s)\kappa, \rho_1, \dots, \rho_{e-1})$ has entries in the specified range, as mean to $\rho' \in [0 \dots 2^{l_0+\kappa}]$. That request each ρ_{0i} for $i = 1, \dots, e - 1$ satisfied :

$$|\rho_{0i}| + 2^l \cdot \kappa_{max} \leq 2^{l_0+\kappa}$$

where $\kappa_{max} = \max\{|a| : a \text{ is an entry in some sweeping vector}\}$

So the probability that a ρ_{0i} is not good is :

$$1 - \frac{2^{l_0+\kappa} - 2^l \cdot \kappa_{max}}{2^{l_0+\kappa}} = \frac{2^l \cdot \kappa_{max}}{2^{l_0+\kappa}}$$

It follows that the statistical distance between the distribution of A 's shares of s and s' is at most twice the probability that ρ is not good. Which we can estimate by the union bound as $e - 1$ times the probability that a single entry is out of range. So $|s' - s| \leq 2^l$. the distance is at most

$$2 \cdot \frac{2^l \kappa_{max} (e-1)}{2^{l_0+\kappa}} \leq 2^{-k}$$

Now the Theorem 1 holds.

6 Optimization of Our Scheme

In this section, we will introduce combinatorial structure [12,14] into our scheme. The combinatorial structure provides a predetermined arrangement of the servers which permits the possibility of reducing the computation of the scheme. This method optimizes our scheme apparently.

6.1 Set Systems

A set system is a pair (X, \mathcal{B}) , where X is a set of n points and \mathcal{B} is a collection of subsets of X called blocks.

We will use a set system with the following properties:

1. $|B| \geq t$ for any $B \in \mathcal{B}$;
2. For any subset $F \subseteq X$ with $|F| \leq b$, there exists a $B \in \mathcal{B}$ such that $F \cap B = \emptyset$. where $t \leq \frac{n}{4} - 1$

It is easy to see that such a set system exists.

Definition 6. A collection \mathcal{T} of k -subsets of $\{1, \dots, n\}$ (called blocks) is an (n, k, b) -covering if every b -subset of $\{1, \dots, n\}$ is contained in at least one block.

Theorem 2. A set system \mathcal{T} satisfies above properties:1,2 if and only if the set system

$$\{1, \dots, n\} \setminus T : T \in \mathcal{T}$$

is an $(n, n - t, b)$

It is easy to see that if (X, \mathcal{B}) is an $(n, n - k, b)$ -covering, then the set system

$$\{1, \dots, n\} \setminus T : T \in \mathcal{T}$$

is a set system satisfying our purpose.

6.2 Applying Set System to the Verifiable LISS

The idea of using the set system is to reduce the computations for the share renewal and recover protocol. In the Section 4, share renewal and recover used the data from all the participants. However, these operations can be carried out using the data from good players. So there are redundant computations. On the other hand, we should be very careful when the good players are selected, since the adversary is mobile. The good player could turn to bad at any time. Thus in the scheme of this section, we will actually select correct information instead of good servers, although we will use "good player" for convenience.

Now let us use the set system to improve our LISS scheme. Suppose (X, \mathcal{B}) is a set system satisfying the condition of subsection 4.1, where $X = \{1, \dots, n\}$, and $\mathcal{B} = \{B_1, \dots, B_s\}$. The set system is published so that each participant can consult it.

Note that in our scheme, in any phase there is a list \mathcal{L} containing all the bad players. By the property of the set system, there is a block B which contains only good players. If the system can determine one of the "good" blocks, then the system can renew the shares or recover the shares only using the data from these players. We will call these players as the members of an *executive committee*.

For a list \mathcal{L} of bad players, the system can decide following list of blocks: B_{i1}, \dots, B_{ie} , such that $B_{ij} \cap \mathcal{L} = \emptyset, j = 1, \dots, e$. These blocks are called *executive committee candidates*. Note that the adversary is mobile, therefore we cannot guarantee that these candidates contain only good servers in the next time period.

The verifiable LISS scheme with combinatorial structure works. In each time period the system dose the following:

1. Run the *Detection* to obtain the list \mathcal{L} of bad players and the executive committee candidates: B_{i1}, \dots, B_{ie} ;

2. If an *executive committee* has not been found, then for next *executive committee candidates* B , each $P_k \in B$ dose the following:

- (1) Each P_k selects a random symmetric matrix $\rho^k = (\rho_0^k, \dots, \rho_{e-1}^k)$, where $\rho_{ij}^k = \rho_{ji}^k$, and $\rho_{00}^k = 0$, M is matrix of ISP. We send $M\rho_0^k$ as shares, $M\rho_i^k, i = 1, \dots, e - 1$ as public verifiable vectors;

- (2) P_k sends $M_{\psi(k)}\rho_0^k$ to $P_m, m = 1, \dots, n$. P_m checks whether $M_{\psi(k)}\rho^k M_{\psi(m)} = M_{\psi(m)}\rho^k M_{\psi(k)}$, If the conditions are not satisfied, P_m broadcasts an accusation of P_k .

- (3) A member in B is accused by at least $b + 1$ players is bad. If a member in B is accused by at most b players, then it can defend itself. If no member in B is bad, then B is found to be the *executive committee*.

3. The system runs the *recovery* protocol to recover the shares for the players in \mathcal{L} ;
4. Each player $P_k \in B$ updates its shares:

$$M_{\psi(k)}\rho_0^k + M_{\psi(k)}\rho_0$$

The reconstruction protocol is the same as in the Section 3.

6.3 Performance Evaluation

In this subsection, we analyze the efficiency of our scheme more clearly and explicitly. We claim the scheme is more efficient, according to the two reasons as follows:

First reason, the traditional Verifiable secret sharing scheme used exponentiation with public key, the cost of computation is tremendous. This paper introduces a proactive verifiable LISS protocol, whose computation is only the level of integral multiplication. So this method improves efficiency in the practical application.

The second reason is that we apply combinatorial structure into our scheme. The combinatorial structure provides a predetermined arrangement of the servers, which reduce the computations for the share renewal and share recover protocol. From comparing the difference and advantage between our proposal and previous scheme, our scheme is more efficient.

7 Conclusion

In this paper, we propose a proactive verifiable Linear Integer Secret Sharing protocol which improves efficiency in the practical application. Then we describe the process of shares renewal and recovery carefully and prove it correct, private and verifiable. We also scheme out a verifiable LISS scheme with combinatorial structure which makes the scheme more efficient. Finally, we give the performance evaluation of this scheme.

Acknowledgement

This work is supported by a grant from the National High Technology Research and Development Program of China (No. 2007AA01Z431). The authors would like to thanks the anonymous referees for their helpful comments.

References

1. Blackley, G.R.: Safeguarding cryptographic keys. In: AFIPS 1979, pp. 313–317 (1979)
2. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
3. Damgard, I., Thorbek, R.: Linear Integer Secret Sharing and Distributed Exponentiation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 75–90. Springer, Heidelberg (2006)

4. Benaloh, J.C., Leichter, J.: Generalized Secret Sharing and Monotone Functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–35. Springer, Heidelberg (1990)
5. Cramer, R., Fehr, S.: Optimal Black-Box Secret Sharing over Arbitrary Abelian Groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 272–287. Springer, Heidelberg (2002)
6. Chor, B., Kushilevitz, E.: Secret Sharing Over Infinite Domains. In: Cryptology 1993, vol. 6(2), pp. 87–95 (1993)
7. Chor, B., Mihály Geréb, G., Kushilevitz, E.: Private Computations over the Integers. *SIAM J. Comput.* 24(2), 376–386 (1995)
8. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks. In: ACM Symposium on principles of distributed computing 1991, pp. 51–59 (1991)
9. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
10. Feldman, P.: A Practical Scheme of Non-Interactive Verifiable Secret sharing. In: 28th Annual Symp. on the Foundations of Computing Science 1987, pp. 427–437 (1987)
11. Pedersen, T.P.: Non-interactive and information-theoretic secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
12. Rees, R.S., Stinson, D.R., Wei, R., Rees, G.H.J.V.: An application of covering designs: determining the maximum consistent set of shares in a threshold scheme. In: *Ars Combinatoria* 1999. LNCS, vol. 531, pp. 225–237. Springer, Heidelberg (1999)
13. Damgård, I., Thorbek, R.: Linear Integer Secret Sharing and Distributed Exponentiation (full version). The Eprint archive, <http://www.iacr.org>
14. Stinson, D.R., Wei, R.: Unconditionally Secure Proactive Secret Sharing Scheme with Combinatorial Structures. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 200–214. Springer, Heidelberg (2000)
15. Alon, N., Gail, Z., Yung, M.: Efficient dynamic-resharing “verifiable secret sharing” against mobile adversary. In: Spirakis, P.G. (ed.) ESA 1995. LNCS, vol. 979, pp. 523–537. Springer, Heidelberg (1995)
16. Boppana, R.B.: Amplification of Probabilistic Boolean Formulas. In: *Advances in Computing Research* 1989, pp. 27–45 (1989)
17. Santis, A.D., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: STOC 1994, pp. 522–533 (1994)
18. Shoup, V.: Practical Threshold Signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
19. Gordon, D.M., Kuperberg, G., Patashnik, O.: New constructions for covering design. *J. Combin. Designs* 3, 269–284 (1995)
20. Stinson, D.R.: *Cryptography Theory and Practice*. CRC Press, Inc., Boca Raton (1995)
21. Ghodosi, H., Pieprzyk, J.: Cheating Prevention in Secret Sharing. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 328–341. Springer, Heidelberg (2000)

A Multi-stage Secret Sharing Scheme Using All-or-Nothing Transform Approach

Mitra Fatemi*, Taraneh Eghlidos, and Mohammadreza Aref

Sharif University of Technology,
Information and System Security Laboratory (ISSL)
Azadi Ave., Tehran, Iran
m.fatemi@ee.sharif.edu
{teghlidos, aref}@sharif.edu

Abstract. A multi-stage secret sharing (MSS) scheme is a method of sharing a number of secrets among a set of participants, such that any authorized subset of participants could recover one secret in every stage. The first MSS scheme was proposed by He and Dawson in 1994, based on Shamir’s well-known secret sharing scheme and one-way functions. Several other schemes based on different methods have been proposed since then. In this paper, the authors propose an MSS scheme using All-Or-Nothing Transform (AONT) approach. An AONT is an invertible map with the property that having “almost all” bits of its output, one could not obtain any information about the input. This characteristic is employed in the proposed MSS scheme in order to reduce the total size of secret shadows, assigned to each participant. The resulted MSS scheme is computationally secure. Furthermore, it does not impose any constraint on the order of secret reconstructions. A comparison between the proposed MSS scheme and that of He and Dawson indicates that the new scheme provides more security features, while preserving the order of public values and the computational complexity.

Keywords: Multi-stage secret sharing, All-or-nothing transforms, Resilient functions.

1 Introduction

In order to provide both security and availability for a given secret, one way is to distribute it among a number of shareholders (participants). The distribution should be accomplished in such a way that any subset of participants, the size of which is at least equal to a given number, be able to reconstruct the secret, using their shares (shadows). More specifically, a (t, n) -threshold secret sharing scheme refers to the procedure of assigning each of the n participants a private share, such that every subset of at least t participants could recover the secret. This concept was introduced by Shamir [1] and Blakley [2] in 1979, independently.

* This work is partially supported by ITRC under the contraction no. T500/20961 and supported by cryptography chair of Iran NSF.

Later, various features like verifiability of the shares [3], resistance against the presence of a number of cheaters [4], dynamic change of the threshold and the number of participants [5] were added to the threshold secret sharing scheme.

A (t, n) -threshold secret sharing scheme is called *perfect* if less than t participants neither could reconstruct the secret, nor obtain any information about it. It has been shown that in a perfect secret sharing scheme, the size of the shares should at least be the same as the size of the secret [6]; in the case of equality, the scheme is referred to as *ideal*. Now, suppose that there are more than one secret to be shared among a group of participants. The dealer may run a perfect threshold secret sharing scheme for each of the secrets and send the related shares to each of the participants via a secure channel. In this way, even though the problem is solved, the difficulty of managing the possible large number of shadows arises. That is, each participant needs to keep multiple shadows to participate in each run of the secret sharing. Besides, protecting shares from an unauthorized access becomes more difficult, due to the increase in the number of shares assigned to each shareholder.

The concept of multi-stage secret sharing (MSS) was introduced by He and Dawson in 1994 [7] to solve the above mentioned problem. An MSS scheme is a method of distributing many secrets among a set of participants, while each of them gets only one master shadow. In an MSS scheme, the secrets are recovered one by one in different stages, possibly according to a pre-specified order. MSS schemes usually need a number of public values. The shareholders, who wish to participate in a secret reconstruction stage, derive the corresponding sub-shadow from their master-shadow and these public values.

The MSS scheme in [7] was based on the Shamir's polynomial secret sharing, a one-way (hash) function and the concept of "shift values". Later, various MSS schemes were presented based on different methods such as coding approach [8], congruence relations and Chinese Remainder Theorem [10] and linear equations [9]. The computational complexity and number of public values are two important factors for comparing the efficiency of MSS schemes and a number of publications appeared to reduce the value of these parameters [11] and [12].

In this paper, the authors employ *All-or-Nothing Transforms (AONT)* to realize the concept of an MSS scheme. An AONT is an invertible and randomized transformation \mathbb{T} , which reveals no information about x even if *almost all* the bits of $\mathbb{T}(x)$ are known. This concept was first introduced by Rivest [13] and further improved by Canetti et al. [14]. In the proposed MSS scheme, AONT is utilized so as to dramatically reduce the size of secret shadows corresponding to a particular secret. Therefore, one could share more secrets among the participants by assigning each of them several private values (shadows), the total size of which is equal to the size of each secret. Regarding the information theoretic lower bound of shares proposed in [15] and [6], an unconditionally secure MSS scheme is impossible. More Precisely, to achieve an unconditionally secure (perfect) secret sharing scheme, the size of each shadow should be at least equal to the sum of the sizes of different secrets. This contradicts the definition of the MSS schemes, which are proposed to reduce the size of the shares. Hence, the

proposed scheme aims to provide the computational security. The authors compare the new MSS scheme with that of He and Dawson. This comparison shows that the new scheme provides more security features, while the computational complexity and number of public values remain almost the same. Moreover, the secrets should be reconstructed according to a pre-specified order in [7], while they can be recovered with an arbitrary order in the proposed scheme.

The rest of the paper is organized as follows. In section [2], we briefly review the MSS scheme proposed by He and Dawson. In section [3] the concept of AONT is described and some of its features are illuminated. The authors propose the new MSS scheme in section [4]. A thorough analysis of the new scheme together with a comparison between the proposed scheme and that of He and Dawson is presented in the subsequent section. Finally, a summary of the whole paper is given in section [6].

2 He and Dawson’s MSS Scheme

In this section, we briefly explain the earliest MSS scheme proposed by He and Dawson.

Let p be an odd large prime number. All the values in this scheme are chosen from the field $GF(p)$. Let s_1, s_2, \dots, s_m denote m secrets to be shared according to (t, n) -threshold schemes among n participants. Suppose that $f : GF(p) \rightarrow GF(p)$ is a one-way function. For any x and any nonnegative integer k , $f^k(x)$ resembles k successive application of f to x . Let x_1, x_2, \dots, x_n be the public identities of the n participants. The dealer performs the following steps.

1. Choose n random values y_1, y_2, \dots, y_n and privately send $y_i, i = 1, \dots, n$, to the i th participant as his/her shadow.
2. For $j = 1, \dots, m$, choose a random polynomial of degree $t - 1$ with the constant value equal to s_j :

$$g_j(x) = s_j + a_{1,j}x + a_{2,j}x^2 + \dots + a_{t-1,j}x^{t-1} \tag{1}$$

and compute $g_j(x_i)$ and $d_{i,j} = g(x_i) - f^{j-1}(y_i)$ for every $1 \leq i \leq n$. The values $d_{i,j}$ are called shift values. Publish all shift values.

The secrets reconstruction process should be conducted in m successive stages with j th secret s_j reconstructed at the $(m-j+1)$ th stage. When a shareholder p_i wants to participate in the j th secret reconstruction stage, he/she should submit the value $g_{m-j+1}(x_i)$ which can be calculated by adding the sub-shadow $f^{m-j}(y_i)$ to the public value $d_{i,m-j+1}$. Having t points on the function $g_{m-j+1}(x)$, one could obtain s_{m-j+1} . The pre-specified order of secret reconstruction in this scheme is needed to guarantee that no information leaks about the shadows corresponding to the undisclosed secrets from the revealed ones.

A security analysis of this scheme is given in section [5.2].

3 All-or-Nothing Transforms

The concept of AONT was first introduced in [13]. However, more general descriptions of it and its applications are presented in [14].

Definition 1. *An l -AONT is a randomized polynomial time computable transformation $\mathbb{T} : \{0, 1\}^k \rightarrow \{0, 1\}^s \times \{0, 1\}^p$ such that [14]:*

- *For any string x , given (all the bits of) $\mathbb{T}(x)$, one can efficiently recover x .*
- *Any polynomial time adversary that learns all but l bits of the secret part of $\mathbb{T}(x)$, obtains “no information” about x , where the first s bits of the output indicates the secret part and the last p bits of it represents the public part.*

Indeed, all-or-nothing transforms allow to encode any x in such a way that the encoding is easily invertible, and still, an adversary who learns all but l bits of the secret part of the encoded data, cannot extract any useful information about x . Therefore, using AONT, we can protect an arbitrary secret x by storing $\mathbb{T}(x)$ in an “exposure-resilient” way. That is, even if almost all the bits of $\mathbb{T}(x)$ are exposed, no information can be revealed about x .

AONT could be categorized into three classes: AONT with perfect security, AONT with statistical security and AONT with computational security (for an exact definition, see [14]). However, the last class of AONT involves parameters taken from a wider range of values. In addition, in an MSS scheme, we look for computational security rather than perfect security. Hence, this class of AONT is taken into consideration hereafter.

There are many approaches towards devising AONT, some of which are applying a hash function to the message, using a scheme based on an FFT-like arrangement of randomized multi-permutations and an approach based on secret sharing schemes [13]. Another construction for an AONT $\mathbb{T} : \{0, 1\}^k \rightarrow \{0, 1\}^s \times \{0, 1\}^k$ is presented in [14] in which the process of creating an l -AONT is seen as that of a one-time-pad encryption. Here, the encryption of $x \in \{0, 1\}^k$ is just $x \oplus R$, where R is a random string of length k derived by inserting an l -exposure resilient function $f : \{0, 1\}^s \rightarrow \{0, 1\}^k$ on a secret value r , that is $R = f(r)$. An l -exposure resilient function is a concept tightly related to l -AONT, which means that knowing all but l bits of the input, no one could gain any information about the output [16, 14]. The l -AONT output is the pair $(r, x \oplus f(r))$. The following theorem [14] has been obtained from this construction and it will be used in the design of the proposed MSS scheme.

Theorem 1. *Assume $l \leq s \leq \text{poly}(l)$ where $m = \text{poly}(k)$ indicates that m is polynomially bounded in k . There exist functions $\mathbb{T} : \{0, 1\}^k \rightarrow \{0, 1\}^s \times \{0, 1\}^k$ (with secret output of length s and public output of length k), such that \mathbb{T} is a computationally secure l -AONT with $l < k \leq \text{poly}(s)$.*

A reasonable setting seems to be $s = O(k)$ (that is just slightly smaller than k) and $l = \lfloor s^\epsilon \rfloor$ to have excellent exposure-resilience [14].

4 The Proposed AONT Approach to Multi-Stage Secret Sharing

In this section, we propose an MSS scheme based on l -AONT with computational security. The value of parameters in the new scheme are derived from the results of theorem [□](#).

4.1 Reducing the Share Size in a Secret Sharing Scheme

Before considering the special case of an MSS scheme, here we explain how one could reduce the size of shares in a secret sharing scheme, using an AONT. To clarify the technique, the well-known Shamir’s threshold secret sharing scheme is utilized. However, it could be implemented on any other perfect secret sharing scheme. The security level of the resulted scheme depends on the secrecy of the AONT; that is, the scheme would have statistical or computational security if an AONT with the same level of security is used. As stated before, we just focus on computationally secure ones.

All secret sharing protocols consist of two processes: distribution and reconstruction. In the distribution process, a dealer shares a secret and distributes the shares among participants. In the secret reconstruction process, the shareholders exchange their shares and reconstruct the secret. In the following, we describe each process for the secret sharing scheme with reduced share size.

:Let S denotes a secret that the dealer wishes to share among a set $\{p_1, \dots, p_n\}$ of n participants according to a (t, n) -threshold secret sharing scheme. Let p be a large prime number such that $S \in GF(p)$. The dealer first chooses $t - 1$ arbitrary random coefficients a_1, a_2, \dots, a_{t-1} , uniformly distributed over $GF(p)$, and constructs the following polynomial:

$$f(x) = S + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1} \tag{2}$$

Next, he/she chooses n distinct nonzero values $x_1, \dots, x_n \in GF(p)$ as the identities corresponding to the n participants and calculates $y_1 = f(x_1), \dots, y_n = f(x_n)$, which are values in $GF(p)$. Suppose that all elements in $GF(p)$ have size k , that is, they could be represented by k bits, where $k = \lceil \log_2 p \rceil$. The dealer chooses a computationally secure l -AONT $\mathbb{T} : \{0, 1\}^k \rightarrow \{0, 1\}^s \times \{0, 1\}^k$ with $s = O(k)$ and $l = s^\epsilon$ for some small $0 < \epsilon < 1$. Then he/she computes the values $\mathbb{T}(y_1), \mathbb{T}(y_2), \dots, \mathbb{T}(y_n)$ and publishes the $k + s - l$ least significant bits of each (least significant $s - l$ bits of the secret part and all k bits of the public part). Finally, the dealer sends the l most significant private bits of $\mathbb{T}(y_1), \mathbb{T}(y_2), \dots, \mathbb{T}(y_n)$ to the n participants as their secret shadows and publishes the map \mathbb{T} and its inverse.

: Let a subset of t participants $\{p_{i_1}, \dots, p_{i_t}\}$ come together in order to reconstruct the secret. By appending the corresponding $(k + s - l)$ -bit public values to their l -bit shadows, the shareholders could obtain the whole bits of $\mathbb{T}(y_{i_1}), \mathbb{T}(y_{i_2}), \dots, \mathbb{T}(y_{i_t})$. Next, they could efficiently recover y_{i_1}, \dots, y_{i_t} , using the inverse transform \mathbb{T}^{-1} . Having t points on the secret polynomial of degree

$t - 1$, the participants are able to calculate the polynomial and recover the constant coefficient S .

Note that in the above scheme, the size of each share is l/k times of the secret size which is a small fraction, due to the choice of l -AONT parameters. Indeed, $l/k \lesssim s^\epsilon/s$ for some small $0 < \epsilon < 1$. Below, a multi-stage secret sharing scheme for m secrets is represented where m denotes $\lfloor k/l \rfloor$.

4.2 Sharing Multi Secrets

Let $S_1, \dots, S_m \in GF(p)$ denote the m secrets to be shared among the participants p_1, \dots, p_n , according to (t, n) -threshold secret sharing schemes. The dealer performs all steps mentioned in the distribution process of section 4.1 for each secret. Let $y_{i,j} = f_j(x_i)$ for $1 \leq i \leq n$ and $1 \leq j \leq m$, where $f_j(x)$ is the polynomial used for sharing the j th secret. In this case, the shadow of the i th shareholder is the concatenation of the first l bits taken from each of $\mathbb{T}(y_{i,1}), \mathbb{T}(y_{i,2}), \dots, \mathbb{T}(y_{i,m})$. Likewise, the public values are comprised of the last $k + s - l$ bits of each $\mathbb{T}(y_{i,j})$, for $1 \leq i \leq n$ and $1 \leq j \leq m$.

Now, let a group of t participants $\{p_{i_1}, \dots, p_{i_t}\}$ wish to collaborate on reconstructing the secret S_j in a stage. Each of them first attaches those l private bits from his/her shadow, which are related to the j th secret, to the corresponding $k + s - l$ bits of the public values. In this way, they could recover the values $\mathbb{T}(y_{i_1,j}), \mathbb{T}(y_{i_2,j}), \dots, \mathbb{T}(y_{i_t,j})$. Then, using the inverse map, the cooperating participants are able to derive the values $y_{i_1,j}, y_{i_2,j}, \dots, y_{i_t,j}$. Finally, using the Lagrange interpolation, the participants could recover the secret S_j .

$$f_j(x) = \sum_{u=1}^t y_{i_u,j} \prod_{v=1, v \neq u}^t \frac{x - x_{i_v}}{x_{i_u} - x_{i_v}} = S_j + a_{1,j}x + a_{2,j}x^2 + \dots + a_{t-1,j}x^{t-1} \quad (3)$$

In the proposed scheme, the total size of each shadow assigned to a participant is $m \times l$ bits which is $m \times l/k$ times of the secret size. Regarding that $m = \lfloor k/l \rfloor$, we have $m \times l/k = \lfloor k/l \rfloor \times l/k \lesssim 1$. This implies that the share size does not exceed the secret size. Also, the total size of public values in this scheme is $m \times (k + s - l) \times n$ bits, which is $\lfloor k/l \rfloor \times (k + s - l)/k \times n \lesssim \lfloor k/l \rfloor \times (2k - l)/k \times n = \lfloor k/l \rfloor \times (2 - l/k) \times n \lesssim (2m - 1) \times n$ times of the secret size.

It is easy to check that in the proposed scheme, there is not any constraint on the order of secret reconstruction and the participants could disclose the secrets at any order they wish. Moreover, the threshold corresponding to the various secrets could be different. Indeed, it suffices to make use of secret polynomials with different degrees. A comprehensive analysis of the proposed MSS scheme and a comparison with that of He and Dawson is presented in the next section.

5 Investigation of the Proposed Scheme

In this section, we discuss about the security and performance of the proposed scheme in two parts. In the first part, it is shown that the scheme provides

computational security. In the subsequent part, efficiency of the scheme is investigated and a comparison with some of other MSS schemes, especially that of He and Dawson's scheme is made. The reason behind this choice of the reference scheme is due to the simplicity of its structure. Moreover, it is the first scheme which introduced the concept of multi-stage secret sharing [7]. The comparison results show that the proposed scheme achieves two additional security features, while preserving the same order of computational complexity and public values.

5.1 Security Analysis

So as to demonstrate that the proposed scheme provides computational security, we state the following two theorems.

Theorem 2. *In the proposed scheme, a group of participants whose number is less than the threshold t , do not gain any information about any of the secrets $S_j, 1 \leq j \leq m$.*

Proof. To prove this assertion, we assume that there are at most $t - 1$ participants who conspire to discover the secret S_j in one stage. To achieve this goal, the collaborating participants need to recover t points of $f_j(x)$ (as defined in [3]). However, they have at most $t - 1$ points of it. Since the secret polynomial coefficients are randomly chosen from $GF(p)$ with a uniform distribution, the secret takes all the values in $GF(p)$ with the same probability when the unknown point of $f_j(x)$ varies over $GF(p)$ (p is a large prime number). Hence, the cheaters (collaborating participants) obtain no information about the secret. From the other side, since $\mathbb{T}(x)$ is a computationally secure l -AONT, knowing at most $k + s - l$ bits from $\mathbb{T}(x)$, makes it computationally infeasible to gain any information about x . Hence, the public values do not leak any information about the shares of non-attendant participants.

Theorem 3. *In the proposed scheme, there is no information leakage from reconstructed secrets to the non-disclosed ones.*

Proof. Since all coefficients of every secret polynomial (including the secrets) are chosen uniformly at random from $GF(p)$, the shares generated by one of them are independent from those generated by the others. Hence, there is no information leakage from the reconstructed secret(s) and the revealed shares to the non-disclosed ones. The two above theorems ensure that the proposed scheme offers the desired level of security.

5.2 Comparative Results

Here, we compare the proposed MSS scheme with that of He and Dawson from the following points of view: Number of public values and share size, the computational complexity of the scheme, and security features.

Before investigating the special case of He and Dawson's scheme [7], we should remark that the schemes in [8] and [12] focus on reconstructing the secrets simultaneously. However, the scheme proposed in [7] and that of this paper have the

privilege of recovering the secrets in different stages. Hence, it does not seem reasonable to compare these two types of multi-secret sharing schemes. In addition, in [10] there is an increase in the share size as a consequence of applying the Chinese Remainder Theorem on different sub-shadows. Indeed, in this scheme, the share size is equal to the sum of the secret sizes.

The comparison results between the proposed MSS scheme and [7] are presented in Table 1. It could be inferred from the results that the share size in both schemes is nearly the same as the secret size. Note that if the number of shared secrets in the proposed scheme be less than $m = \lfloor k/l \rfloor$, the share size would be smaller than the secret size. Besides, the number of public values of the proposed scheme has the same order as in [7]. Precisely speaking, it is two times of the number of public values in the scheme presented in [7].

The scheme proposed by He and Dawson employs a one-way function in addition to the Shamir’s threshold secret sharing scheme. However, the proposed MSS scheme based on l -AONT approach utilizes an all-or-nothing transform together with the Shamir’s secret sharing scheme. The construction of the applied AONT is based on a one-time pad encryption and a resilient function. Also, resilient functions have structures based on one-way functions [14]. As a consequence, both schemes make use of similar structures with the same number of times (Table 1).

Table 1. Comparison of the proposed scheme with that of He & Dawson

	He-Dawson’s scheme	Prposed scheme
share size to secret size ratio	1	$m \times l/k \approx 1$
No. of public values	$m \times n$	$(2m - 1) \times n$
Computational complexity	$m \times$ Shamir’s scheme $(m - 1) \times n$ one-way function	$m \times$ Shamir’s scheme $m \times n$ AONT
Shadow memory	$m \times n$	n

In the MSS schemes proposed in [7] and [17], once a number of bits of a participant’s master-shadow reveals, the security of all of his her sub-shadows gets compromised. This is a consequence of deriving different sub-shadows from a master-shadow. This problem is inhibited in our scheme by deriving different sub-shadows independently.

As a final point, we indicate that in the He and Dawson’s scheme, once a participant receives his/her master shadow y_i , he/she has to compute all sub-shadows $f(y_i), f^2(y_i), \dots, f^{m-2}(y_i), f^{m-1}(y_i)$ since $f^{m-1}(y_i)$ is supposed to be the first sub-shadow he/she would use. This is implicitly equivalent to an increase in the share size. That is, each shareholder needs $m \times k$ bits of memory (shadow memory) to store m units of k -bit sub-shadow. The larger the share size, the more susceptible shares to the information leakage. On the other side, once a sub-shadow is

exposed, all of the subsequent sub-shadows, derived from it, get revealed. This is resulted from applying a one-way function to the participants master shadows in order to derive their different sub-shadows. The proposed scheme brings this problem to an end by independently generating the sub-shadows.

6 Conclusions

In this paper, the authors have considered secret sharing schemes with several secrets and proposed a new approach, based on l -AONT, for multi-stage secret sharing schemes. Under these functions, any bit string is converted to an exposure-resilient one, that is, having “almost all” bits of the output, one could not obtain any information about the input. Using this property, the authors have reduced the share size such that the total size of sub-shadows assigned to a participant for reconstructing different secrets has become as small as a secret size. To the best of author’s knowledge, this is the first time that l -AONTs (resilient functions) are used to realize secret sharing schemes. The proposed MSS scheme is compared with that of He and Dawson. The results indicate that the new scheme has removed the security drawbacks in their scheme, which are resulted from deriving different sub-shadows by applying a one-way function on the previous sub-shadow. Still, the number of public values and the computational complexity of the scheme have the same order as those of [7].

References

1. Shamir, A.: How to Share a Secret. *Commun. ACM* 22(11), 612–613 (1979)
2. Blakley, G.R.: Safeguarding Cryptographic Keys. In: *AFIPS, National Computer Conference*, vol. 48, pp. 313–317 (1979)
3. Stadler, M.: Publicly Verifiable Secret Sharing. In: Maurer, U.M. (ed.) *EUROCRYPT 1996*. LNCS, vol. 1070, pp. 190–199. Springer, Heidelberg (1996)
4. Ogata, W., Kurosawa, K.: Optimum Secret Sharing Scheme Secure against Cheating. In: Maurer, U.M. (ed.) *EUROCRYPT 1996*. LNCS, vol. 1070, pp. 200–211. Springer, Heidelberg (1996)
5. Barwick, S.G., Jackson, W.A., Martin, K.M.: Updating the Parameters of a Threshold Scheme by Minimal Broadcast. *IEEE Transactions on Information Theory* 51(2), 620–633 (2005)
6. Stinson, D.R.: An Explication of Secret Sharing Schemes. *Des., Codes, Cryptogr.* 2, 357–390 (1992)
7. He, J., Dawson, E.: Multi-Stage Secret Sharing Scheme Based on One-way Function. *Electronic Letters* 30(19), 1591–1592 (1994)
8. Chien, H.Y., Jan, J.K., Tseng, Y.M.: A Practical (t, n) Multi-Secret Sharing Scheme. *IEICE Transactions on Fundamentals* E83-A(12), 2762–2765 (2000)
9. Runhua, S., Liusheng, H., Yonglong, L., Hong, Z.: A Threshold Multi-Secret Sharing Scheme. In: *IEEE International Conference on Networking, Sensing and Control, ICNSC 2008*, pp. 1705–1707 (2008)
10. Chan, C.W., Chang, C.C.: A Scheme for Threshold Multi-Secret Sharing. *Applied Mathematics and Computation* 166, 1–14 (2006)

11. Harn, L.: Comment: Multistage Secret Sharing based on One-way Function. *Electronics Letters* 31(4), 262 (1995)
12. Yang, C.C., Chang, C.C., Hwang, M.S.: A (t, n) multi-secret sharing scheme. *Applied Mathematics and Computation* 151(2), 483–490 (2004)
13. Rivest, R.: All-or-Nothing Encryption and the Package Transform. In: Biham, E. (ed.) *FSE 1997*. LNCS, vol. 1267, pp. 210–218. Springer, Heidelberg (1997)
14. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-Resilient Functions and All-or-Nothing Transforms. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
15. Karnin, E.D., Greene, J.W., Hellman, M.E.: On Secret Sharing System. *IEEE Transaction on Information Theory* 29(1), 35–41 (1983)
16. Chor, B., Friedman, J., Goldreich, O., Hastad, J., Rudich, S., Smolensky, R.: The Bit Extraction Problem or t -resilient Functions. In: *FOCS*, pp. 396–407 (1985)
17. He, J., Dawson, E.: Multisecret-Sharing Scheme Based on One-way Function. *Electronic Letters* 31(2), 93–95 (1995)

Digital Audio Watermarking Technique Using Pseudo-Zernike Moments

Xiangyang Wang¹, Tianxiao Ma¹, and Panpan Niu²

¹ School of Computer & Information Technology, Liaoning Normal University,
Dalian 116029, China

² School of Information Science & Technology, Dalian Maritime University,
Dalian, 116026, China

{wxy37, tianxiao87, niupanpan333}@126.com

Abstract. Based on Pseudo-Zernike moments and synchronization code, we propose a new digital audio watermarking algorithm with good auditory quality and reasonable resistance toward de-synchronization attacks in this paper. Simulation results show that the proposed watermarking scheme is not only inaudible and robust against common signals processing such as MP3 compression, noise addition, re-sampling, and re-quantization etc, but also robust against the de-synchronization attacks such as random cropping, amplitude variation, pitch shifting, etc.

1 Introduction

Due to the advent of network and computer technology, there has been an explosion in growth of the use of digital media through electronic commerce and on-line services. Since digital media is easily reproduced and manipulated, anyone is potentially capable of incurring considerable financial loss. Digital watermarking is introduced to safeguard against such loss. [1].

Nowadays, there is an unprecedented development in the audio watermarking field. On the other hand, attacks against audio watermarking systems have become more sophisticated [2]. In general, these attacks can be categorized into common signal processing and de-synchronization attacks. Most of the previous audio watermarking schemes are robust to common signal processing, but show severe problems to de-synchronization attacks. Fortunately, several approaches against the de-synchronization attacks have been developed in recent years. These schemes [2] can be roughly divided into exhaustive search, invariant watermark, self-synchronization, and synchronization pattern.

Exhaustive search: In [3] and [4], by performing multiple correlation tests, the authors applied the detection engine to search for resynchronization. However, exhaustive search schemes need large amount of calculation, and often cause false alarm. *Invariant watermark:* Mansour et al. [5-6] proposed a time-scale invariant watermarking embedding strategy by changing the relative length of the middle segments between two successive maximum peaks of the smoothed waveform. In [7],

by using music content analysis, the authors presented a new audio watermarking method. The watermark is robust to pitch-invariant TSM but vulnerable to the common signal processing. *Self-synchronization*: In [8], self-synchronization was implemented by applying special peak point extraction scheme. In general, the current self-synchronization algorithm cannot extract invariant audio feature steadily. *Synchronization pattern*: In [9], the authors chose Bark code which has better self-relativity as synchronization mark and embedded it into temporal domain. However, this method is vulnerable to some de-synchronization attacks such as amplitude variation, pitch shifting, jittering, time-scale modification (TSM) etc.

Pseudo-Zernike moment is an ideal region-based shape descriptor, with the characteristics of rotation invariant, low noise sensitive, expression effectiveness and fast computation. It has been widely used for pattern recognition, signal analysis and other fields [10]. Based on Pseudo-Zernike moment and synchronization code, we propose a new digital audio watermarking algorithm. The watermark bit is embedded into the average value of modulus of the low-order Pseudo-Zernike moment. Meanwhile combining the two adjacent synchronization code searching technology, the algorithm can extract the watermark without the help from the original digital audio signal.

2 Fundamental Theory and Synchronization

2.1 Fundamental Theory

In our audio watermarking scheme, the watermark can be embedded into the host audio by 3 steps. Firstly, the original digital audio is segmented and then each segment is cut into two parts. Secondly, with the spatial watermarking technique, synchronization code is embedded into the statistics average value of audio samples in the first part. And then, map 1-D digital audio signal in the second part into 2-D form, and calculate its Pseudo-Zernike moments. Finally, the watermark bit is embedded into the average value of modulus of the low-order Pseudo-Zernike moments.

2.2 Synchronization Code

Synchronization is one of the key issues of audio watermarking. Watermark detection starts by alignment of watermarked block with detector. Losing synchronization causes false detection. Time-scale or frequency-scale modification makes the detector lose synchronization. So we need exact synchronization algorithms based on robust synchronization code.

Generally, we should avoid false synchronization during selecting synchronization code. Several reasons contribute to false synchronization: (1) the style of the synchronization code, (2) the length of synchronization code, (3) the probability of "0" and "1" in synchronization code. Among of them, the length of synchronization code is especially important. The longer it is, the more robust it is.

The proposed scheme embeds Barker code in front of the watermark to locate the position where watermark is embedded. Barker codes, which are subsets of PN sequences, are commonly used for frame synchronization in digital communication systems. Barker codes have low correlation side lobes. A correlation side lobe is the correlation of a codeword with a time-shifted version of itself.

3 The Pseudo-Zernike Moments

Pseudo-Zernike moments consist of a set of complex polynomials [11] that form a complete orthogonal set over the interior of the unit circle, $x^2 + y^2 \leq 1$. If the set of these polynomials is denoted by $\{V_{nm}(x, y)\}$, then the form of these polynomials is as follows

$$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho) \exp(jm\theta) \tag{1}$$

where $\rho = \sqrt{x^2 + y^2}$, $\theta = \tan^{-1}(y/x)$. Here n is a non-negative integer, m is restricted to be $|m| \leq n$ and the radial Pseudo-Zernike polynomial $R_{nm}(\rho)$ is defined as the following

$$R_{nm}(\rho) = \sum_{s=0}^{n-|m|} \frac{(-1)^s (2n+1-s)! \rho^{n-s}}{s!(n+|m|+1-s)!(n-|m|-s)!} \tag{2}$$

Like any other orthogonal and complete basis, the Pseudo-Zernike polynomial can be used to decompose an analog 2-D signal $f(x, y)$

$$f(x, y) = \sum_{n=0}^{\infty} \sum_{\{m: |m| \leq n\}} A_{nm} V_{nm}(x, y) \tag{3}$$

where A_{nm} is the Pseudo-Zernike moment of order n with repetition m . Given a 2-D signal of size $M \times N$, its Pseudo-Zernike moments (approximate version) are computed as

$$\hat{A}_{nm} = \frac{n+1}{\pi} \sum_{i=1}^M \sum_{j=1}^N h_{nm}(x_i, y_j) f(x_i, y_j) \tag{4}$$

where the value of i and j are taken such that $x_i^2 + y_j^2 \leq 1$, and

$$h_{nm}(x_i, y_j) = \int_{x_i - \frac{\Delta x}{2}}^{x_i + \frac{\Delta x}{2}} \int_{y_j - \frac{\Delta y}{2}}^{y_j + \frac{\Delta y}{2}} V_{nm}^*(x, y) dx dy \tag{5}$$

where $\Delta x = \frac{2}{M}$, $\Delta y = \frac{2}{N}$, $h_{nm}(x_i, y_j)$ can be computed to address the nontrivial issue of accuracy. In this research, we adopt the following formulas (6) which are most commonly used in literature to compute Pseudo-Zernike moments of discrete 2-D signals, and the orthogonality and completeness of the Pseudo-Zernike yield the following formula (7) for reconstructing the 2-D signal.

$$\hat{A}_{nm} = \frac{n+1}{\pi} \sum_{i=1}^M \sum_{j=1}^N V_{nm}^*(x_i, y_j) f(x_i, y_j) \Delta x \Delta y \tag{6}$$

$$\hat{f}(x, y) = \sum_{n=0}^{n_{\max}} \sum_{m=-n}^n A_{nm} V_{nm}(x, y) \tag{7}$$

4 Analysis of Audio Pseudo-Zernike Moments

In this paper, the digital watermark will be embedded into original audio signal by utilizing the excellent characteristics of Pseudo-Zernike moments. However, in transmission process, audio signal (or watermarked audio) may suffer from attacks such as low-pass filtering, MP3 compression, and amplitude variation etc, which will impact the audio Pseudo-Zernike moments unavoidably. So, it is very necessary to analyze the audio Pseudo-Zernike moments and select the stable audio Pseudo-Zernike moments for embedding.

4.1 Decomposition and Reconstruction of Audio Pseudo-Zernike Moments

Digital audio signal is one-dimensional (1-D) discrete signal, and the two-dimensional (2-D) Pseudo-Zernike transform cannot be performed on digital audio directly, so the 1-D digital audio $\{g(i)\}$ must be mapped into a 2-D form $\{f(x, y)\}$ by using the following

$$\begin{cases} L = R \times R + M, & 0 \leq M < 2R + 1 \\ f(x, y) = g(x \cdot R + y), & 0 \leq x, y \leq R \end{cases} \quad (8)$$

where $f(x, y)$ is corresponding 2-D audio version after projection, L is the length of 1-D digital audio, M is the rest of audio samples, and R is the width or height in $f(x, y)$ which should be as large as possible under the constraint of Equation (8).

After mapping, Pseudo-Zernike decomposition and reconstruction procedures on audio signal are performed by using Equation (6) and (7). For the convenience of explaining the audio Pseudo-Zernike decomposition and reconstruction, we choose a clip from our test

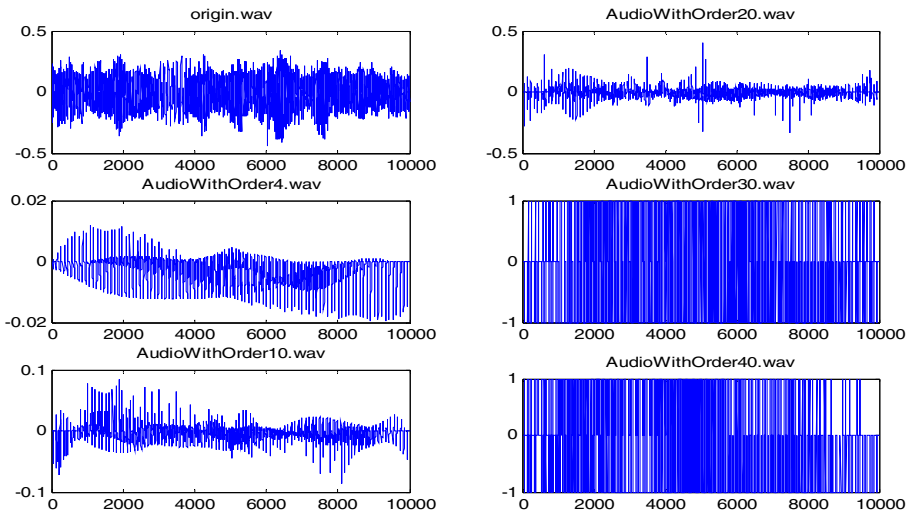


Fig. 1. The original audio and the reconstructed audios under the different order

data set, flute music denoted as *origin.wav* (16-bit signed mono audio file with the length of 1.25s), for testing under different sampling rate from 8 kHz to 44.1 kHz. The number of the given max order N_{\max} is assigned to 4, 10, 20, 30, and 40, respectively. The waveform of original one and the reconstructed audios are aligned in Fig.1 (Here, the sampling rate is 8 kHz, and the audio signal is mapped into 100×100 form).

In Fig. 1, *origin.wav* is the original audio while *AudioWithOrder*.wav* denote the reconstructed ones, in which N_{\max} is assigned as '*'. It is noted that the low-order moments captured the basic shape of audio signal while the higher order ones fill the high frequency details. This observation is similar to that in images [11]. The degradation caused in reconstruction procedure is due to that when N_{\max} is lower the high frequency information is discarded, while N_{\max} is higher the cumulative computation error occurs in the reconstruction [11]. Referred to Fig.1, it is evident that the reconstruction degradation from limited moments is unavoidable. As to other kinds of audio, such as pop music, piano music and speech, etc., the simulation results are similar.

4.2 Selection of Audio Pseudo-Zernike Moments

In the following experiment, we investigate the robustness performance of audio Pseudo-Zernike moments to common signal processing. We choose a clip flute music denoted as *music.wav* (16-bit signed mono audio file sampled at 44.1 kHz with 250000 audio samples), for testing. First, the digital audio is mapped into 500×500 form. And then, the following mathematical expression is designed to compute the modification of moments before and after audio processing.

$$E_{an} = \sum \|Z_{nm}\|, E_{bn} = \sum \|Z_{nm}''\| \quad (9)$$

where $\|Z_{nm}\|$, $\|Z_{nm}''\|$ are the modulus of Pseudo-Zernike moments of order n ($0 \leq n \leq N_{\max}$) with repetition m . E_{an} and E_{bn} denote the total amplitude of all moments with the given order n before and after audio processing, respectively.

1) The influence of low-pass filtering on audio Pseudo-Zernike moments

Fig. 2 shows the influence of low-pass filtering on audio Pseudo-Zernike moments under different cutoff frequency. We can see that the Pseudo-Zernike moments under order 20, with cutoff frequency of 0.8 kHz, are very robust to low-pass filtering.

2) The influence of MP3 compression on audio Pseudo-Zernike moments

Fig. 3 shows the influence of MP3 compression on audio Pseudo-Zernike moments under different bit rates. We can see that the Pseudo-Zernike moments under order 10, with the lowest bit rate of 32 kbps, are very robust to MP3 compression.

3) The influence of amplitude variation on audio Pseudo-Zernike moments

Fig. 4 gives the influence of amplitude variation on audio Pseudo-Zernike moments (Scaling factor α is 0.8 and 1.2, respectively). We can see that the relation between audio amplitude and its Pseudo-Zernike moments is linear.

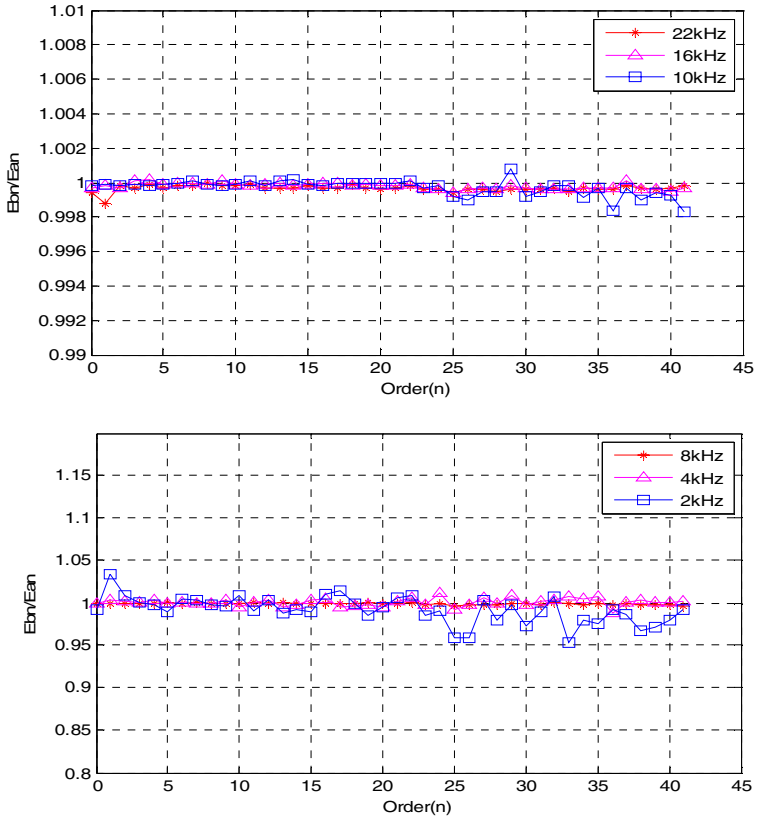


Fig. 2. The effects of low-pass filtering on audio Pseudo-Zernike moments under different cutoff frequency

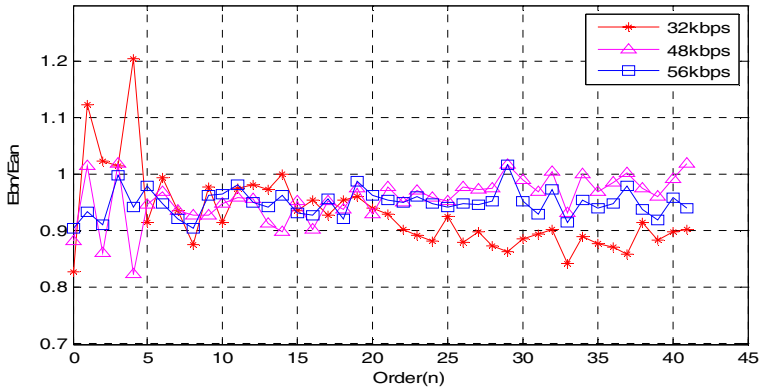


Fig. 3. The effects of MP3 compression on audio Pseudo-Zernike moments under different bit rates

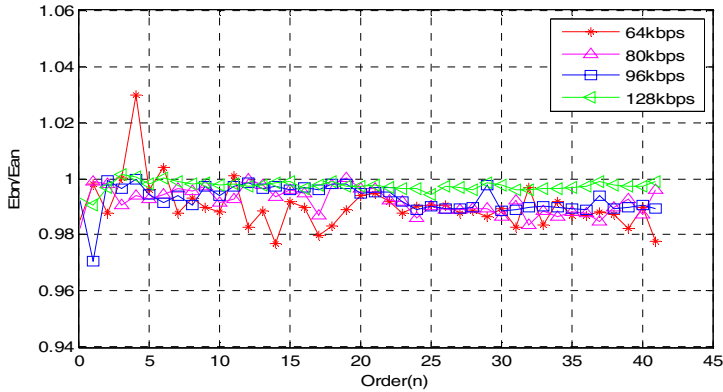


Fig. 3. (Continued)

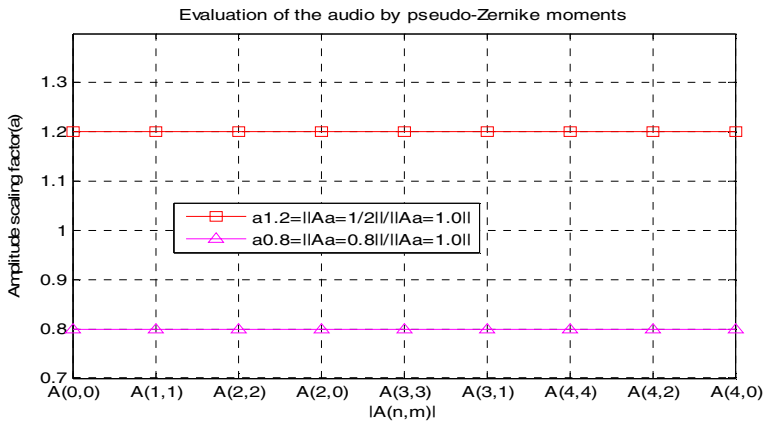


Fig. 4. The relationship between amplitude variation and Pseudo-Zernike moments

We select flute music as the example clip to test the effect of MP3 compression, low-pass filtering and amplitude variation. As to other kinds of audio, such as pop music, piano music and speech, etc., the simulation results are similar. Based on the extensive testing with different audio signals, we have the following observations:

i) Pseudo-Zernike transform of 1-D signal may be achieved by mapping the signal into 2-D form. It is noted that the low-order moments capture the basic shape of the signal but the reconstruction degradation from Pseudo-Zernike moments is large and unavoidable.

ii) Based on the extensive experiments, it is also found that the low-order Pseudo-Zernike moments are robust to common signal processing. The moments under order 10 are very robust to MP3 compression even with the lowest bit rate of 32 kbps. The moments under order 20 are robust to low-pass filtering up to with cutoff frequency of 0.8 kHz.

iii) Scaling linearly audio amplitude not only can change the audio Pseudo-Zernike moments, but also is insensitive to human perceptual system. The relation between audio amplitude and its Pseudo-Zernike moments is linear.

As a conclusion, if we embed the digital watermark into that Pseudo-Zernike moments under order 10 and try to avoid the degradation in reconstruction procedure, it is expected that the watermark will be very robust to common signal processing and some hostile attacks.

5 Watermark Embedding Scheme

We propose a new audio watermarking scheme in which the Pseudo-Zernike moments and synchronization code are utilized. Firstly, the original audio is segmented and then each segment is cut into two parts. Secondly, with the spatial watermarking technique, synchronization code is embedded into the statistics average value of audio samples in the first part. And then, map 1-D digital audio signal in the second part into 2-D form, and calculate its Pseudo-Zernike moments. Finally, the watermark bit is embedded into the average value of modulus of the low-order Pseudo-Zernike moments.

Let $A = \{a(i), 0 \leq i < Length\}$ represent a host digital audio signal with *Length* samples.

$W = \{w(i, j), 0 \leq i < M, 0 \leq j < N\}$ represent a binary image to be embedded within the host audio signal, and $w(i, j) \in \{0,1\}$ is the pixel value at (i, j) .

$F = \{f(i), 0 \leq i < Lsyn\}$ represent a synchronization code with *Lsyn* bits, where $f(i) \in \{0,1\}$.

The main steps of the embedding procedure based on Pseudo-Zernike moments and synchronization code can be described in detail as follows.

5.1 Watermark Preprocessing

In order to dispel the pixel space relationship of the binary watermark image, and improve the robustness of the whole digital watermark system, watermark scrambling algorithm is used at first. In our watermark embedding scheme, the binary watermark image is scrambled from W to W_1 by using Arnold transform, where Then, it is transformed into a 1-D sequence of ones or zeros, and each bit of the watermark data is mapped into an antipodal sequence using BPSK modulation:

$$W_1 = \{w_1(i, j), 0 \leq i < M, 0 \leq j < N\}$$

$$W_2 = \{w_2(k) = w_1(i, j), 0 \leq i < M, 0 \leq j < N, k = i \times N + j, w_2(k) \in \{0,1\}\} \tag{10}$$

$$W_3 = \{w_3(k) = 1 - 2 \times w_2(k), k = 0, 1, \dots, M \times N - 1, w_3(k) \in \{-1,1\}\}$$

In order to improve the robustness of proposed scheme, audio segmenting is used at first. Then, each segment is cut into two parts with L_1 and L_2 samples, respectively, where

$$A(i) = \{a(iL + k), 0 \leq k < L\} \quad (0 \leq i < \lfloor \frac{Length}{L} \rfloor) \tag{11}$$

Where $L = L_1 + L_2$, $L_1 = L_{syn} \times n$, n is a constant and is chosen to be 5 samples in our experiment, $L_2 = R \times R \times M \times N \times 9$ (We use 9 sub-segments to embed 1 watermark bit).

5.2 Synchronization Code Embedding

In order to guarantee robustness and transparency of watermarking, the proposed scheme embeds synchronization code into the statistics average value of audio samples as follows.

1) The first part A_1^0 of audio segment A^0 is cut into L_{syn} audio sub-segments, and each audio sub-segment $PA_1^0(m)$ having n samples, where

$$PA_1^0(m) = \{pa_1^0(m)(i) = a_1^0(i + m \times n), 0 \leq i < n, 0 \leq m < L_{syn}\}$$

2) Calculating the average value of $PA_1^0(m)$:

$$\overline{PA_1^0(m)} = \frac{1}{n} \sum_{i=0}^{n-1} pa_1^0(m)(i), (0 \leq m < L_{syn})$$

3) The synchronization code can be embedded into each $PA_1^0(m)$ by quantizing the average value $\overline{PA_1^0(m)}$, the rule is given by

$$pa_1^{\prime 0}(m)(i) = pa_1^0(m)(i) + (\overline{PA_1^{\prime 0}(m)} - \overline{PA_1^0(m)})$$

where $PA_1^0(m) = \{pa_1^0(m)(i), 0 \leq i < n\}$ is original sample, and $PA_1^{\prime 0} = \{pa_1^{\prime 0}(m)(i), 0 \leq i < n\}$ is modified sample.

5.3 Watermark Embedding

1) The second part A_2^0 of audio segment A^0 is cut into audio sub-segments and each audio sub-segment $A_2^0(k)$ ($k = 0, 1, \dots, M \times N \times 9 - 1$) is chosen to have $R \times R$ samples.

2) The audio sub-segment $A_2^0(k)$ is mapped into 2-D form, and low-order Pseudo-Zernike moments are calculated. Then, the total modulus of the low-order Pseudo-Zernike moments is given

$$S_k = \sum_{n=0}^{N_{max}} \sum_m \|Z_{nm}\|$$

where S_k is the total modulus of the Pseudo-Zernike moments for the k^{th} audio sub-segment $A_2^0(k)$, Z_{nm} is the low-order Pseudo-Zernike moments of order n with repetition m . And in this paper, $N_{max} = 10$.

3) Three successive sub-segments are selected as a group (sub-segment group), and the average value of modulus of the low-order Pseudo-Zernike moments for the r^{th} Sub-segment group is

$$AVE_r = \frac{S_{k-1} + S_k + S_{k+1}}{3} \tag{12}$$

4) Let the average values of modulus in the three consecutive sub-segment groups be represented as AVE_{r-1} , AVE_r and AVE_{r+1} . Their relations may be obtained from the following Equation

$$\begin{cases} A = E_{\max} - E_{\text{med}} \\ B = E_{\text{med}} - E_{\min} \end{cases} \tag{13}$$

where A and B stand for the differences, respectively. And

$$E_{\max} = \text{Max}(AVE_{r-1}, AVE_r, AVE_{r+1})$$

$$E_{\min} = \text{Min}(AVE_{r-1}, AVE_r, AVE_{r+1})$$

So we can exploit the following Equation to embed one digital watermark bit $w(i)$

$$\begin{cases} A - B \geq T & \text{if } w_3(i) = 1 \\ B - A \geq T & \text{if } w_3(i) = -1 \end{cases} \tag{14}$$

where $T = d \cdot \frac{(AVE_{r-1} + AVE_r + AVE_{r+1})}{3}$ is the embedding strength, and d is the intensity factor.

If the Equation (14) holds, we will embed the watermark bit $w(i)$ directly; else we use the strategy as followed in Table I to adjust E_{\max} , E_{med} and E_{\min} until they satisfy Equation (14).

5) The sub-segments are reconstructed by using the modified Pseudo-Zernike moments. Assumed that after embedding one digital watermark bit, AVE_{r-1} , AVE_r and AVE_{r+1} go to AVE'_{r-1} , AVE'_r and AVE'_{r+1} , respectively. It is equivalent to scale AVE_{r-1} , AVE_r and AVE_{r+1} by using the corresponding factor α_{r-1} , α_r and α_{r+1} , which may be computed by the following expressions

$$\alpha_{r-1} = \frac{AVE'_{r-1}}{AVE_{r-1}} \quad \alpha_r = \frac{AVE'_r}{AVE_r} \quad \alpha_{r+1} = \frac{AVE'_{r+1}}{AVE_{r+1}} \tag{15}$$

According to the analysis of audio Pseudo-Zernike moments, we can know that the relationship between audio amplitude and its Pseudo-Zernike moments is linear (see Section 4.2, Fig.4). It means that the modification of Pseudo-Zernike moments may be mapped as the operation of scaling audio amplitude. Using this conclusion, we

Table 1. The adjust Strategy of $E_{max}, E_{med}, E_{min}$

<p>If embedded watermark bit $w_3(i)$ is '1' and A-B<T: Step 1: Emax increase. Step 2: If (Emed>Emin) Emed reduce, Emin increase. Else if (Emed<=Emin & Emin >0) Emed reduce, Emin reduce.</p>	<p>If embedded watermark bit $w_3(i)$ is '-1' and B-A<T: Step 1: If (Emin >0) Emin reduce. Step 2: If (Emax>Emed) Emax reduce □ Emed increase. Else if (Emax<=Emed) Emax increase, Emed increase.</p>
---	--

introduce the following strategy to generate the watermarked audio by scaling the sample values in each sub-segment, referred to Equation (16).

$$\begin{cases} f'_{k-1}(x, y) = \alpha_r \cdot f_{k-1}(x, y) \\ f'_k(x, y) = \alpha_r \cdot f_k(x, y) \\ f'_{k+1}(x, y) = \alpha_r \cdot f_{k+1}(x, y) \end{cases} \quad (16)$$

where α_r is the amplitude scaling factor of the r^{th} audio sub-segment group (It has three successive sub-segments, $f_{k-1}(x, y)$, $f_k(x, y)$, and $f_{k+1}(x, y)$), computed by using Equation (14). $f_k(x, y)$ and $f'_k(x, y)$ denote the k^{th} sub-segment of the original 2-D signal and the watermarked 2-D signal, respectively.

5.4 Repeat Embedding

In order to improve the robustness against cropping, the proposed scheme repeats 5.2 and 5.3 sections to embed synchronization code and watermark into every audio segment. Finally, by using Equation (8) we obtain the reconstructed watermarked audio A' .

6 Watermark Detecting Scheme

The watermark detecting procedure in the proposed scheme neither needs the original audio signal nor any other side information. The synchronization code detection and digital watermark extraction are two key steps in the whole watermark detection procedure.

6.1 Synchronization Code Detection

Synchronization code detection refers to check the synchronization code in the audio data segment covered by the window (the size is L_1), and synchronization code detection can be described as follows.

1) According to Section 5.2, the average value $PA^*(m)$ of former $n \times m$ audio samples $PA^*(m)$ in the audio data segment (covered by the window) is calculated.

$$PA^*(m) = \{pa^*(m)(i) = a^*(i + m \times n), 0 \leq i < n, 0 \leq m < Lsyn\}$$

2) The synchronization code is extracted by using below rule

$$F' = \{f'(m) = \left\lfloor \frac{PA^*(m)}{S_1^*} \right\rfloor \bmod 2, \quad 0 \leq m < L_{syn}\}$$

where S_1^* is the quantization step size.

3) In order to avoid effectively false synchronization, the frame synchronization technology of digital communications (The bit comparison) is utilized for identifying the synchronization code. That is to say, if the extracted synchronization code is same completely as the original one, the synchronization code is thought to be found, and the corresponding position is recorded.

6.2 Digital Watermark Extraction

In this paper, digital watermark is extracted from the audio data segment between two adjacent synchronization codes.

1) The audio data segment A_2^{0*} (the size is L_2^*), which is candidate audio segment for watermark extraction, is defined by two adjacent synchronization codes.

2) The audio data segment A_2^{0*} is cut into audio sub-segments and each audio sub-segment $A_2^{0*}(k)$ is mapped into 2-D form. Then, the low-order Pseudo-Zernike moments are calculated. See Section 5.3.

3) As in Equation (12), we compute AVE_{r-1}^* , AVE_r^* and AVE_{r+1}^* , which are ordered to obtain E_{max}^* , E_{med}^* and E_{min}^* . Similar to Equation (13), we have

$$\begin{cases} A^* = E_{max}^* - E_{med}^* \\ B^* = E_{med}^* - E_{min}^* \end{cases}$$

Comparing A^* and B^* , we can get the hidden watermark bit by using the following rule

$$w_3^*(i) = \begin{cases} 1 & \text{if } A^* - B^* \geq 0 \\ -1 & \text{if } A^* - B^* < 0 \end{cases}$$

The process is repeated until all hidden watermark bits are extracted.

4) Step 1)-step 3) is repeated until several copies of watermark are extracted, and the optimal digital watermark $W_4^* = \{w_4^*(i)\}$ ($i = 0, 1, \dots, M \times N - 1$) is obtained according to the majority rule.

5) The 1-D binary sequences W_2^* is obtained by BPSK demodulating the optimal watermark W_4^*

$$W_2^* = \{w_2^*(k) = (1 - w_4^*(k)) / 2, \quad k = 0, 1, \dots, M \times N - 1, w_2^*(k) \in \{0, 1\}\}$$

6) The 1-D binary sequences W_2^* are rearranged to form the binary watermark image W_1^* .

7) Finally, the watermark image

$$W^* = \{w^*(i, j), 0 \leq i < M, 0 \leq j < N\}$$

can be obtained by descrambling.

7 Experimental Results

In order to evaluate the performance of our scheme, performance test and robustness test are illustrated for the proposed watermarking algorithm. All of the audio signals in the test are music with 16 bits/sample, 44.1kHz sample rates, and 20 seconds. We use a 16x16 binary image as our watermark for all audio signals and a 16-bit Barker code 1111100110101110 as synchronization code. The quantization step $S_1 = 0.2$, the intensity factor $d = 0.2$, the audio sub-segment is mapped into $R \times R = 8 \times 8$ 2-D form, and $N_{max} = 8$.

In order to illustrate the robust nature of our watermarking scheme, common signal processing and de-synchronization attacks are used to estimate the robustness of our scheme, as shown in Table 2. The PSNR of proposed algorithm is 40.39dB.

Table 2. The watermark detection results for various attacks (BER)

Attack free	Re-quantization	Re-sampling (22.05kHz~8kHz)	Low-pass filtering (9kHz-3kHz)	Low-pass filtering (6kHz)
0	0	0	0	1.75
MP3 (256kb~56kb)	Equalization	Noise addition	Echo addition	Low-pass filtering (2kHz)
0	0	0	6.25	4.47
Cropping (1s~6s)	Adding (1s~2s)	Pitch shift one degree higher	Pitch shift one degree lower	Amplitude-scaling (180%~10%)
0	0	0	0	0
TSM (+1%)	TSM (-1%)	TSM (-2%)	TSM (-3%)	TSM (-4%)
38.28	46.48	46.09	50.78	53.52
Low-pass filtering(4kHz) + Cropping 1s	Re-sampling (11.025kHz) + Low-pass filtering (8kHz)	Re-sampling (22.05kHz) + Cropping 1s	Low-pass filtering(8kHz) + Amplitude -scaling 150%	Noise addition + Amplitude -scaling 50%
0	0	0	0	0
Noise addition + MP3(112k)	Re-quantization + Noise addition	Equalization + Cropping 1s	Re-quantization + Adding 1s	Noise addition + Cropping 1s
0	0	0	0	0

8 Conclusion

De-synchronization attacks are the Achilles heel for many audio watermarking schemes. Based on Pseudo-Zernike moments and synchronization code, we propose a new digital audio watermarking algorithm with good auditory quality and reasonable resistance toward de-synchronization attacks in this paper. Simulation results show that the proposed watermarking scheme is not only inaudible and robust against common signals processing but also robust against the de-synchronization attacks. In addition, the watermark can be extracted without the help of the original digital audio signal and can be easily implemented.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 60773031 & 60873222, the Open Foundation of State Key Laboratory of Networking and Switching Technology of China under Grant No. SKLNST-2008-1-01, the Open Foundation of State Key Laboratory of Information Security of China under Grant No. 03-06, the Open Foundation of State Key Laboratory for Novel Software Technology of China under Grant No. A200702, the Open Foundation of Key Laboratory of Modern Acoustics Nanjing University under Grant No. 08-02, and Liaoning Research Project for Institutions of Higher Education of China under Grant No. 2008351.

References

- [1] I. J. Cox, L. M. Matthew, A. B. Jeffrey, et al. Digital watermarking and steganography, Second Edition, Burlington, MA, Morgan Kaufmann Publishers (Elsevier), 2007.
- [2] N. Cvejic, T. Seppänen. Digital audio watermarking techniques and technologies: applications and benchmarks, Information Science Reference, Hershey, PA, USA, 2007.
- [3] D. Kirovski, H. Malvar. Robust covert communication over a public audio channel using spread spectrum. In Proceedings of the 4th International Workshop on Information Hiding (IH 2001), 2001, Lecture Notes in Computer Science 2137, Springer 2001: 354-368.
- [4] D. Kirovski, H. Malvar. Spread-spectrum watermarking of audio signals. IEEE Trans. on Signal Processing, 2003, 51(4): 1020-1033.
- [5] M. Mansour, A. Tewfik. Audio watermarking by time-scale modification. In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2001, 3: 1353-1356.
- [6] M. Mansour, A. Tewfik. Data embedding in audio using time-scale modification. IEEE Trans. on Speech Audio Processing, 2005, 13(3): 432-440.
- [7] W. Li, X. Xue. Localized audio watermarking technique robust against time-scale modification. IEEE Trans. on Multimedia, 2006, 8(1): 60-69.
- [8] Xiaohong Ma, Bo Zhang, Xiaoyan Ding. Self-synchronization blind audio watermarking based on feature extraction and subsampling. International Symposium on Neural Networks 2007, Lecture Notes in Computer Science 4492, 2007:40-46.
- [9] X. Y. Wang, W. Qi, P. P. Niu. A new adaptive digital audio watermarking based on support vector regression. IEEE Trans. On Audio, Speech and Language Processing, 2007, 15(8): 2270-2277.

- [10] J. Haddadnia, M. Ahmadi, K. Faez. An efficient feature extraction method with Pseudo-Zernike moments in RBF neural network-based human face recognition system. EURASIP Journal on Applied Signal Processing, 2003, (9): 890-901.
- [11] Javad Haddadnia, Majid Ahmadi, and Karim Faez. An efficient feature extraction method with Pseudo-Zernike moments in RBF neural network-based human face recognition system. EURASIP Journal on Applied Signal Processing, 2003, 3(9): 890-901.

An Image Sanitizing Scheme Using Digital Watermarking

Masatoshi Noguchi¹, Manabu Inuma², Rie Shigetomi²,
and Hideki Imai^{1,2}

¹ Faculty of Science and Engineering, Chuo University, Tokyo, Japan
{masatoshi-noguchi}@imailab.jp

² Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
{imuma.manabu,rie-shigetomi,h-imai}@aist.go.jp

Abstract. We developed an efficient and secure image sanitizing scheme that enables integrity check and authentication of a sanitized image. Sanitizing is to delete or conceal sensitive data such as national secrets or personal information in digital images/documents. With a typical digital signature or digital watermarking scheme, a sanitized image cannot be authenticated because the integrity of the content is broken when it is sanitized. Our image sanitizing scheme solves this problem by combining a digital signature, digital watermarking and error correction technique with Reed-Solomon (RS) code. In addition, due to the RS encoding, we can realize the lightweight image sanitizing with small signature size, which is quite significant for the practical implementation. Furthermore, we investigate the security of our image sanitizing scheme. Considering the security requirements for image sanitizing, our scheme turned out to be quite effective to safely release a sensitive image to the public. The detailed explanation of our image sanitizing scheme and its security evaluation are given in this paper.

1 Introduction

When digital documents or images are published in the Internet, guaranteeing their integrity and authenticity is essential to prevent malicious activities such as impersonation and falsification. For such purpose, digital signature and digital watermarking are widely used in various applications. In addition, when digital contents are released to the public, sensitive information, for example personal data and national secrets, should be deleted or concealed for the privacy and security. The data processing to delete/conceal sensitive information is called *sanitizing*. Sanitizing, however, causes a problem that the sanitized content is not correctly authenticated since the integrity of the data is broken when it is sanitized. A naive use of a digital signature or digital watermarking cannot achieve both content sanitizing and authentication. Note that generating a digital signature after sanitizing is not the solution because the time stamp of the original signature cannot be restored.

This problem is first pointed out in [1] as *the digital document sanitizing problem*, and several studies tackling this problem have been reported so far [1,2,3,4,5]. As for *the image sanitizing problem*, only a few studies have been reported as far as the authors know [6,7]. In this paper, we propose an efficient and secure image sanitizing scheme combining the digital signature, digital watermarking and Reed-Solomon (RS) code. The previous study [7] only uses a digital watermark and RS code, and therefore, the authenticity of the embedder of the watermark is not considered and the watermark can possibly be fabricated. Our scheme solves this vulnerability by utilizing a digital signature.

In this paper, we explain the detailed model and procedure of the proposed sanitizing scheme. To discuss the security of our scheme, we also present the categories of the attacks against sanitized data and security requirements for the image sanitizing. Then we clarify the feasibility and effectiveness of our sanitizing scheme by comparing its performance with the past studies.

The rest of this paper is organized as follows. Section 2 describes the outline of the related works and their problems. Section 3 explains our image sanitizing scheme and gives the model of our scheme, assumption of the attackers, and security requirements for image sanitizing. Section 4 evaluates the security of our sanitizing scheme, and finally Section 5 summarizes our study.

2 Related Work

2.1 Digital Signature-Based Document Sanitizing

The sanitizing problem of digital content has been mainly studied in *document* sanitizing. Some studies have applied a digital signature technique to authenticating sanitized digital documents [1,2,3,4,5]. These sanitizing schemes are the three-party model that consists of a signer, sanitizer(s) and verifier. The signer generates a signature of the document with a secret key. Then, the documents and signature are transferred from the signer to the sanitizer. The sanitizer divides the document into sanitize-allowed blocks (hereinafter SA) and sanitize-prohibited blocks (hereinafter SP), and sanitizes some of the SAs if necessary. Note that simply masking the SA with random data will destroy the integrity of the document and consequently the authentication will fail. Therefore, special mask data for document sanitizing must be calculated in advance. Due to this mask data, the past sanitizing studies suffer from a large amount of data to transfer and its long computation time. Then finally, the sanitized image and the signature are transferred to the verifier. The verifier checks the digital signature and accepts the image only if the authenticity of the image is successfully verified.

Miyazaki et al. proposed the digital document sanitizing scheme which can control the disclosure condition [2]. The disclosure condition is the information about which blocks of the document are SAs and which are SPs. In this scheme, an SA includes original document and its pre-calculated mask data, while an SP has only the original document. By deleting the pre-calculated mask data, the SA is unable to be sanitized and thus changed to the SP. Miyazaki et al. improved this scheme by using the aggregate signature based on bilinear maps [3].

This scheme can hide the number of the sanitized blocks. However, the digital signature must be recalculated whenever the disclosure condition is updated.

Izu et al. proposed the sanitizable signature scheme where verifiers can authenticate the sanitizer, and thus, the malicious sanitizers can be eliminated [4]. However, this scheme also suffers from the inefficiency due to the requirement of pre-calculated mask data. As described above, the past studies have been suffering from the large data size and long computation time since they must always handle the large data set: the original document, digital signature and pre-calculated mask data. Therefore, simply applying the digital signature for document sanitizing is neither efficient nor practical.

2.2 Digital Watermark-Based Image Sanitizing

Digital watermarking is a technique to secretly embed information in an image. The principle of the digital watermarking is that, even if secret information is scattered in an image, one cannot detect the subtle change of the image. Therefore, the digital watermark cannot be utilized for document sanitizing. The digital watermarking can be applied to authentication and is practically used for protecting the copyright of digital publications and so on. If the integrity of the watermark is broken, one cannot prove that the publication surely belongs to the author. Therefore, the digital watermark should be tamper-resistant for the purpose of authentication.

To the contrary, for tamper-detecting purpose, the digital watermark should be fragile [8,9,10,11]. If the embedded watermark is broken, the image turns out to be tampered. In this paper, we refer to the fragile digital watermark as a tamper-detecting watermark. Kawadu et al. proposed the digital watermark-based image sanitize scheme utilizing fragile watermarks [6,7]. Their scheme is the three-party model which consists of an embedder of the digital watermark, sanitizer(s) and verifier.

The scheme proposed in [6] embeds tamper-detecting watermarks into each SPs. Verifying the watermarks, one can detect tampering with the SAs and SPs, and fabrication of the SPs. This scheme, however, cannot detect fabrication of the SAs. Moreover, this scheme requires the information about which blocks are SAs/SPs, and therefore, it is not efficient for the practical implementation. The scheme proposed in [7] embeds symbols of RS code as a tamper-detecting watermark. Using RS code, the verifier does not need to know the position of SAs/SPs. This scheme, however, does not utilize a digital signature and cannot detect fabrication of the watermark. Additionally in [6,7], the model of the scheme is ambiguous and the security issues are not well considered.

3 The Details of the Image Sanitizing Scheme

3.1 System Model

Our image sanitizing scheme includes four types of players: Client, Embedder, Sanitizer(s) and Verifier. Each players is described as follows:

Client is the person who wants to release a digital image to the public. As shown in Fig. 1, Client divides the image into several blocks, and each of the blocks is assigned as a **sanitize-allowed block (SA)** or **sanitize-prohibited block (SP)**. Though determining SAs and SPs in advance of sanitizing seems a strong restriction, our scheme works quite well in practice since only limited blocks are assigned as SAs.

Embedder generates a digital signature of the image, encodes the signature using RS encoding, and embeds the code in the image as a digital watermark. Since data in SA can be changed by sanitizing, the signature is generated only from SPs to maintain the consistency of the signature, while the signature is embedded in both SAs and SPs.

Sanitizer sanitizes SAs inappropriate for disclosing to the public. When sanitizing the SA, all the data bits are changed into 0. Note that any additional mask data for sanitizing is not required in our scheme, while the pre-calculated mask is required in the previous studies.

Verifier authenticates the transferred image by extracting and verifying the watermark embedded in the image. Note that the verifier does not need to know which blocks are SAs/SPs.

These four types of players are not necessarily the different persons. For example, Client and Embedder can be the same person.

3.2 Attack Models

To concentrate on the security issues on sanitizing scheme, we assume that the channels among the players are secure. Insecure channels should be protected using some proper technology, e.g., PKI. We assume that the client and embedder are honest, and the sanitizer(s) and verifier can be dishonest. In this model, we define the anticipated attacks as follows:

Recovery attack is to restore the original image from the sanitized image, embedded watermark and so on. If the original image is restored by an attacker, the attack is considered as succeeded.

Alteration attack is to alter the image data in SPs. If an SP is altered and the alteration is not detected by Verifier, the attack is considered as succeeded. Altering the data in SAs is legitimate and is not considered as an attack.

Forgery attack is a trial to generate a counterfeit SP that would pass the verification. If the counterfeit SP is not detected by Verifier and passes the verification, the attack is considered as succeeded.

3.3 Security Requirements

Here we describe the security requirements of image sanitizing necessary for safely releasing an image to the public. In our scheme, we introduce three security requirements: Privacy, Inalterability and Unforgeability; the previous work

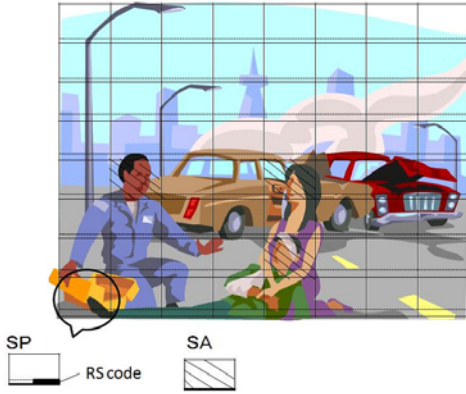


Fig. 1. Example of an image and its SA/SP blocks

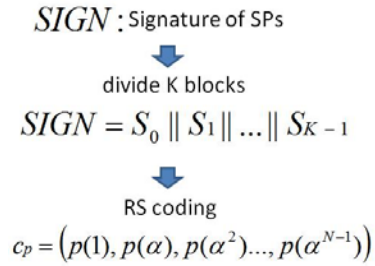


Fig. 2. The procedure for Reed-Solomon (RS) encoding

proposed Invisibility instead of Inalterability [3]. Invisibility is the requirement that the verifier cannot determine the number of the sanitized blocks. The requirement Invisibility, however, constrains the intuitive image sanitizing which we are aiming at, and consequently, Invisibility is not adopted in our scheme. The security requirements in our scheme are summarized as follows:

Privacy: The original image is not restored from the sanitized one.

Inalterability: Altering the data in SPs can be detected.

Unforgeability: The verifier can verify whether the transferred image is correctly sanitized, in other words, the verifier can check if the watermark is generated by the authorized embedder.

3.4 The Detailed Procedure

As described in section 3.1, our image sanitizing scheme includes 4 players: Client, Embedder, Sanitizer and Verifier. First, Client divides the image into several blocks and each block is assigned as an SA or SP. Then Embedder calculates the digital signature from the SPs and embeds it as a watermark into both SAs and SPs. Next, Sanitizer sanitizes some blocks which includes sensitive data, and finally Verifier extracts and verifies the embedded watermark. The detailed procedure is explained as follows.

SA/SP setting

1. Client divides the original image into G blocks.
2. Client designates the sanitize-allowed blocks as SAs, and the rest of the blocks as SPs. The number of the SAs is $G - L$ and the number of the SPs is L .

Watermark embedding

1. Embedder looks for the blocks including secret or sensitive data, and designate such blocks as SAs and others as SPs. Embedder embeds the code symbols (sanitizing watermark) to the determined blocks.
2. Embedder determines the position to embed the watermark for each SA/SP.
3. Embedder divides the signature into $K (= 2L)$ portions.
4. Embedder generates $(K - 1)$ -dimensional information polynomial $p(x)$ from the information bits, and generates an RS code as follows:

$$C_p = \{p(0), p(1), p(\alpha), \dots\}.$$
Here, C_p consists of N code symbols (Fig. 2).
5. Embedder embeds 2 code symbols into the particular position of each SA/SP. In the SA, one of the two symbols is intentionally erroneous; In the SP, both of the 2 symbols are correct. Due to this intentional error, tampering with the SP is surely detected.
6. Embedder embeds another tamper-detecting watermark to the SA separately from the watermark generated using RS code.

Sanitizing

1. Sanitizer determines which SAs should be actually sanitized.
2. Sanitizer sets all pixel values to zero in the selected SA. The image in the SA is blacked out and all information in the SA is deleted.

Verification

1. Verifier extracts the code symbols of RS code from non-sanitized SA/SPs, and decodes the code to obtain the digital signature. If the decode process fails, the image is considered as fraudulently tampered.
2. If the code is correctly decoded, the block including the intentional error symbol is considered as an SA, and others as SPs. Then, Verifier concatenates the SPs(excluding the code symbols), and calculates and verifies the digital signature of the concatenated data.
3. Verifier extracts and verifies the digital watermark from SAs to check if the SAs are tampered.

3.5 Example

Here we consider sanitizing a VGA-size (640×480) image. The size of the divided block is 16×16 , and therefore, there are totally $1200 (= G)$ blocks and $2400 (= 2G = N)$ code symbols in the image. Let r be the number of digits of the code length N . In this example, $r \geq 12$ [bits] because $2^r > N$. Let S be the signature length and here we suppose $S = 1024$. Since the number of the information symbols K must satisfy the inequality $r > S/K$, K must be $K = 2L \geq 86$ because $r \geq 12$. Therefore, the number of the SPs L must be equal to or larger than 43. Presumably, secret or sensitive data are included only in the limited blocks of the image, and thus, the inequality $L \geq 43$ will be satisfied in most cases. To sum up, in this example, two 12-bit symbols (24-bit symbols) are embedded into each SA/SP.

4 Considerations

4.1 Decoding Conditions

Let e be the number of the error symbols and h be the number of the erasure symbols. The decoding condition in RS code is described as follows:

$$2e + h \leq N - K.$$

Assume that the number of sanitized SAs and sanitized SPs are y and y' , respectively. Let $E(= G - L)$ be the number of the error symbols of the pre-sanitized image, then the decode condition is $2(E - y) + 2(y + y') \leq N - K$. After embedding the watermark, since parameters satisfies $2E = 2(G - L) = N - K$, the decode condition is $y' \leq 0$. On the other hand, y' is the number of the sanitized areas and satisfies $y' \geq 0$. Therefore, $p(x)$ can be decoded only when $y' = 0$, that is, $p(x)$ is decoded only when no SP is tampered.

4.2 Evaluation of the Security Requirements

This section evaluates if the proposed scheme satisfies the security requirements described in Section 3.3.

Privacy: Both of the original image data and the watermark in the SA are deleted when the SA is sanitized. Therefore, restoring the original image from sanitized one is impossible.

Inalterability: Tampering with SPs can be detected by verifying the watermark. Tampering with the sanitized SAs can be easily detected since all pixel values are zero in the sanitized SAs. Tampering with non-sanitized SAs can be detected with the sanitizing watermark.

Unforgeability: Since the embedded watermark is the digital signature signed by Embedder, verifying the watermark is equivalent to verifying the digital signature. Therefore, unintended sanitizing by unauthorized persons can be detected by the digital signature scheme.

4.3 Advantage of Signature Encoding

In our scheme, the generated digital signature of SPs is encoded using RS code and embedded to the image, instead of directly embedding the signature without RS coding. The reason for this is that, without encoding, the information about which block is SA/SP is required for extracting the signature. If the information of SA/SP position is tampered, the SA/SP can be substituted. Therefore, the digital signature scheme with RS encoding is resistant against the attack of SA/SP substitution.

4.4 Comparison with Past Studies

Table 1 shows the performance comparison between our image sanitizing scheme and past studies. In Table 1, \bigcirc , \triangle , \times denote the best, middle and worst performance, respectively. As Table 1 shows, our scheme can solve the problems of forgery and SP-alternation in Kawadu's scheme.

Table 1. Comparison with each method

	Sanitizing signature	Kawadu [7]	Our scheme	Signature directly
Efficiency	△	○	○	○
Disclosure condition control	○	×	×	×
Sanitizing demand	×	○	○	○
Privacy	○	○	○	○
SA-inalteration	○	△	△	△
SP-inalteration	○	△	○	○
Unforgeability	○	×	○	○
Substitution of SA/SP	—	○	○	×

5 Conclusions

We developed an efficient and secure digital image sanitizing scheme that enables integrity check and authentication of a sanitized image. Sanitizing is to delete or conceal sensitive data such as national secrets or personal information. With a typical digital signature or digital watermarking scheme, a sanitized image cannot be authenticated because the integrity of the image is broken when it is sanitized. Our image sanitizing scheme solves this problem by combining a digital signature, digital watermarking and error correction technique with Reed-Solomon code. The basic ideas of our scheme is described as follows:

- Divides the image into Sanitize-Allowed (SA) blocks and Sanitize-Prohibited (SP) blocks.
- Generates the digital signature from SP blocks, encodes the signature based on RS code to generate the tamper-detecting codes, and embeds 2 tamper-detecting codes into every SA/SP.
- In SA blocks, one of the 2 tamper-detecting codes is intentionally erroneous. Due to this intentional error, the decoding condition of the RS code is maintained during the image sanitizing.

Additionally, applying RS encoding, we can realize the secure and lightweight image sanitizing with small signature size, and thus, we can utilize the digital watermarking technology to embed the tamper-detecting codes. Therefore, our scheme requires no separate additional data such as a digital signature. This feature is quite significant for the practical use of the proposed scheme. Furthermore, we investigate the security of our image sanitizing scheme in this paper. Considering these security requirements, our scheme is quite effective to safely release a sensitive image to the public.

Acknowledgment

This work was supported partially by a Grant-in-Aid for Scientific Research (19200006) from Japan Society for the Promotion of Science.

References

1. Miyazaiki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshimura, H.: Digital Document Sanitizing Problem. IEICE Technical Report 103(195), 61–67 (2003)
2. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshimura, H., Tezuka, S.: Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 239–246 (2005)
3. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally Signed Document Sanitizing Scheme based on Bilinear Maps. In: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, pp. 343–354 (2006)
4. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: PIATS: A Partially Sanitizable Signature Scheme. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 72–83. Springer, Heidelberg (2005)
5. Izu, T., Kunihiko, N., Ohta, K., Takenaka, M., Yoshioka, T.: A Sanitizable Signature Scheme with Aggregation. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 51–64. Springer, Heidelberg (2007)
6. Kawadu, T., Yoshida, R., Shigetomi, R., Imai, H.: A Sanitizable Technique for Images with Digital Watermarking. In: SCIS Symposium on Cryptography and Information Security (2007) (in Japanese)
7. Kawadu, T., Yoshida, R., Shigetomi, R., Yoshizoe, K., Imuma, M., Imai, H.: A Technique That Sanitize for Digital Watermarking. In: The 29th Symposium on Information Theory and Its Applications (2008) (in Japanese)
8. Barreto, P., Kim, H.Y., Rijmen, V.: Toward Secure Public-key Blockwise Fragile Authentication Watermarking. IEE Proceedings-Vision, Image and Signal Processing 149(2), 57–62 (2002)
9. Suthaharan, S.: Fragile Image Watermarking Using a Gradient Image for Improved Localization and Security. Pattern Recognition Letters 25(16), 1893–1903 (2004)
10. Zhang, X., Wang, S.: Statistical Fragile Watermarking Capable of Locating Individual Tampered Pixels. IEEE Signal Processing Letters 14(10), 727–730 (2007)
11. Lin, C.Y., Sow, D., Chang, S.F.: Using Self-Authentication-and-Recovery Images for Error Concealment in Wireless Environments. In: SPIE ITCOM/OptiComm, vol. 4518, pp. 267–274 (2001)

Adaptive and Composable Oblivious Transfer Protocols (Short Paper)

Huafei Zhu and Feng Bao

I²R, A*STAR, Singapore

Abstract. An adaptive k -out-of- n oblivious transfer protocol ($\text{OT}_{k \times 1}^n$) allows a receiver to obtain $m_{\sigma_{i-1}}$ before deciding on the i -th index σ_i . This paper studies adaptive k -out-of- n oblivious transfer protocols in the presence of static adversaries in the universal composition (UC) framework. We show that the proposed $\text{OT}_{k \times 1}^n$ protocol realizes the UC-security in the $\mathcal{F}_{\text{crs}}^{\mathcal{D}}$ -hybrid model under the joint assumptions that the underlying signature scheme is secure, the decisional Diffie-Hellman problem and the decisional composite residuosity problem in Z_{N^2} are hard as well as all knowledge proof protocols applied in this paper are computational zero-knowledge in the presence of static adversaries.

Keywords: Adaptive security, oblivious transfer, simulator, universal composability.

1 Introduction

An adaptive k -out-of- n oblivious transfer protocol ($\text{OT}_{k \times 1}^n$), first introduced by Naor and Pinkas in the context of the half-simulation model, allows a receiver to obtain $m_{\sigma_{i-1}}$ before deciding on the i -th index σ_i . Naor and Pinkas proposed interesting $\text{OT}_{k \times 1}^n$ protocols in the half-simulation model and showed that their schemes analyzed in the half-simulation model might admit practical attacks on the receiver's privacy [16]. Camenisch, Neven and Shelat [2] and Green and Hohenberger [13] proposed fully-simulatable $\text{OT}_{k \times 1}^n$ protocols under the bilinear assumptions. The proofs of their protocols employ adversarial rewinding, and thus do not support the universally composable security. Green and Hohenberger [14] proposed the first implementation of universally composable adaptive oblivious transfer protocols based on the joint hardness assumptions of symmetric external Diffie-Hellman problem, decision linear problem and q -hidden LRSW problem. A well-motivated problem is thus to find new frameworks for adaptive oblivious transfer schemes in the universally composable framework of Canetti under the standard cryptographic assumptions. At Crypto'08, Peikert, Vaikuntanathan and Waters [18] proposed non-adaptive, universally composable 1-out-of-2 oblivious transfer protocols in the presence of static adversaries. Their protocols are based on a new notion called dual-mode cryptosystem. The dual-mode cryptosystem is set up in one of two modes: extraction mode or decryption mode. A crucial of the dual-mode cryptosystem is that no adversary can distinguish

the common reference string between two modes. The recent work of Kurosara and Nojima [15] provides two weaker simulatable adaptive k -out-of- n oblivious transfer protocols in the standard cryptographic assumptions. However, their proofs invoke the rewinding technique and thus their protocols are not secure in the standard universally composable framework.

To the best of our knowledge, no universally composable adaptive k -out-of- n oblivious transfer protocols constructed from the standard cryptographic assumptions are available in the research community. This paper aims to provide $\text{OT}_{k \times 1}^n$ protocols in the presence of malicious adversaries under the standard cryptographic assumptions. We first set up a verifiably committed database in an initial stage of a protocol execution once the committer gets the instruction (send, sid , $ssid$, S , R) from an environment \mathcal{Z} , where sid is a session id of an $\text{OT}_{k \times 1}^n$ query and $ssid$ is a sub-session id of any j th query in the $\text{OT}_{k \times 1}^n$ execution ($1 \leq j \leq k$). Such a verifiably committed database must be equivocal since the simulator does not know the honest receiver's adaptive index σ_i . This is not the case in the non-adaptive, 1-out-of-2 oblivious transfer protocols. We then ensure that the verifiably public database is extractable during the course of the OT executions since the simulator must extract the malicious sender's input and forward the extracted input to the OT functionality when the environment \mathcal{Z} instructs an adversary \mathcal{A} to send messages to the receiver R on behalf of the corrupted sender S . In a nutshell, we apply the Damgård-Neilsen's mixed commitment protocol [10] to realize the equivocal property and the Bresson-Catalano-Pointcheval's double trapdoor protocol [1] to realize the extractable property.

We will make use of the ideas introduced in [19], [20] and [21] to prove the security. That is, to prove the receiver's security against a malicious sender, the master secret key msk is used to extract implicit input of the corrupted sender. Consequently, a simulator for the corrupted sender is well defined; To prove the sender's security against a malicious receiver, the local keys sk_1, \dots, sk_n are used to extract implicit input of the corrupted receiver and then the simulator makes use of the extractable keys (E-keys) to interpret a fake commitment to a commitment of any message. Such an interpretation is necessary when we consider the adaptive OT protocols where the simulator does not have any knowledge of an honest sender. This technique is crucial to prove the security without involving the standard rewinding technique even though the zero-knowledge proof systems are involved in our implementation.

We stress that we do not apply the corresponding knowledge extractors of zero-knowledge protocols employed in our OT protocol but the double trapdoor mechanism to prove the security of our implementation that is the most significant difference between our ideas and that of Kurosara and Nojima [15]. The application of the double trapdoor mechanism for extracting implicit inputs of corrupted parties is a key point to realize the universally composable security. The Kurosara and Nojima's scheme fails to reach the universally composable security since their proof relies on the notion of the corresponding knowledge extractor of a zero-knowledge proof system where the rewinding technique is unavoidable by the definition of any zero-knowledge proof/argument system.

2 Definitions

Throughout the paper, we assume that the reader is familiar with the following building blocks:

- The Canetti’s universally composable framework [4];
- The Bresson, Catalano and Pointcheval’s double trapdoor encryption scheme [1] which is in turn constructed from the Paillier’s encryption scheme [17];
- The Damgård and Nielsen’s mixed commitments [10];
- The standard zero-knowledge proof techniques such as Camenisch and Michels protocol [3] for proving that N is a product of two safe primes p and q , Damgård-Fujisaki’s protocol [11] for zero-knowledge proof of knowledge of a commitment and the equality of a commitment and an encryption and the Cramer-Damgård-Schoenmakers’ OR-protocol [8].

Common Reference String Model. Canetti and Fischlin have shown that OT cannot be UC-realized without a trusted setup assumption [5]. We thus assume the existence of an honestly-generated Common Reference String (crs) and work in the so called $\mathcal{F}_{\text{crs}}^{\mathcal{D}}$ -hybrid model. The functionality of common reference string model assumes that all participants have access to a common string that is drawn from some specified distribution \mathcal{D} . The functionality defined below is due to [6].

Functionality $\mathcal{F}_{\text{crs}}^{\mathcal{D}}$

$\mathcal{F}_{\text{crs}}^{\mathcal{D}}$ proceeds as follows, when parameterized by a distribution \mathcal{D} .
 When receiving a message (sid, P_i, P_j) from P_i , let $\text{crs} \leftarrow \mathcal{D}(1^n)$ and send (sid, crs) to P_i , and send $(\text{crs}, \text{sid}, P_i, P_j)$ to the adversary, where sid is a session identity. Next when receiving (sid, P_i, P_j) from P_j (and only from P_j), send (sid, crs) to P_j and to the adversary, and halt.

Functionality for adaptive k -out-of- n oblivious transfer $\text{OT}_{k \times 1}^n$. Following [14] and [7], functionality $\text{OT}_{k \times 1}^n$ is described below.

Definition 1. Let \mathcal{F} be a functionality for $\text{OT}_{k \times 1}^n$. A protocol π is said to universally composable realize \mathcal{F} if for any adversary \mathcal{A} , there exists a simulator \mathcal{S} such that for all environments \mathcal{Z} , the ensemble $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ is computationally indistinguishable with the ensemble $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}$.

3 $\text{OT}_{k \times 1}^n$ Based on Bresson-Catalano-Pointcheval’s Protocol

With the help of the Bresson-Catalano-Pointcheval double trapdoor encryption scheme (BCP) and the Damgård-Nielsen’s mixed commitment scheme, we are now ready to describe our $\text{OT}_{k \times 1}^n$ protocol, denoted by π which consists of the

Functionality $\mathcal{F}_{\text{OT}_{k \times 1}^n}$

$\mathcal{F}_{\text{OT}_{k \times 1}^n}$ proceeds as follows, parameterized with κ , k and n ($k \leq n$), and running with an oblivious transfer sender S , a receiver R and an ideal world adversary \mathcal{S} .

- Upon receiving a message $(\text{sid}, \text{sender}, m_1, \dots, m_n)$ from S , where each $m_i \in \{0, 1\}^\kappa$, an imaginary trusted third party (TTP) stores (m_1, \dots, m_n) ;
- Upon receiving a message $(\text{sid}, \text{ssid}_i, \text{receiver}, \sigma_i)$ from R , TTP checks if a $(\text{sid}, \text{sender}, \dots)$ message was previously received and $i \leq k$. If no such message was received, or $i > k$, TTP sends nothing to R . Otherwise, TTP sends $(\text{sid}, \text{ssid}_i, \text{request})$ to S and receives the tuple $(\text{sid}, \text{ssid}_i, b \in \{0, 1\})$ in response.
- TTP then passes $(\text{sid}, \text{ssid}_i, b)$ to the adversary, and: if $b = 0$, TTP sends $(\text{sid}, \text{ssid}_i, \perp)$ to R ; if $b = 1$, TTP sends $(\text{sid}, \text{ssid}_i, m_{\sigma_i})$ to R .

following three subprotocols – a common reference string generation protocol, an initialization protocol and an i th adaptive OT-query protocol.

The common reference string generation protocol. The common reference string generation algorithm is depicted below:

- On input a security parameter λ , the common reference generation algorithm $\text{OTcrsGen}(1^\lambda)$ (for OT) invokes a key generation algorithm \mathcal{G} of the BCP’s encryption to generate composite modulus of the form $N = pq$ that is a product of two safe primes p and q (i.e., $p = 2p' + 1, q = 2q' + 1$), a cyclic group $G \subseteq Z_{N^2}^*$ of order N' ($N' = p'q'$), and n random generators g_1, \dots, g_n of G and n random elements (h_1, \dots, h_n) such that $h_i = g_i^{x_i} \pmod{N^2}, x_i \in_U Z_{N'}, i = 1, \dots, n$. Let $\text{crs}_{\text{DE}} = \langle \text{des}(G), (g_1, h_1), \dots, (g_n, h_n), N \rangle$.
- $\text{OTcrsGen}(1^\kappa)$ then invokes a key generation algorithm of Damgård-Nielsen’s mixed commitment scheme to generate the global public key \tilde{N} and n random keys $K_1, \dots, K_n \in_U Z_{\tilde{N}^2}^*$, where $\tilde{N} = \tilde{p}\tilde{q}, \tilde{p} = 2\tilde{p}' + 1, \tilde{q} = 2\tilde{q}' + 1$, and $\tilde{p}, \tilde{p}', \tilde{q}$ and \tilde{q}' are large prime numbers. Let $\text{crs}_{\text{MC}} = \langle \tilde{N}, K_1, \dots, K_n \rangle$.
- The common reference string generation algorithm also proves that $f(x) = x^N \pmod{N}$ and $\tilde{f}(x) = x^{\tilde{N}} \pmod{\tilde{N}}$ are permutations over Z_N^* and $Z_{\tilde{N}}^*$, respectively. This can be done by invoking Camenisch and Michels protocol that proves in zero-knowledge that the number N (\tilde{N} , respectively) is the product of two safe primes p and q (\tilde{p} and \tilde{q} respectively) [3].
Let ZK-PRoPrimes be a transcript of zero-knowledge to convince a verifier N (\tilde{N} , respectively) are product of large safe primes. Given a copy of ZK-PRoPrimes , one can immediately delivery a proof that $f(x) = x^N \pmod{N}$ ($\tilde{f}(x) = x^{\tilde{N}} \pmod{\tilde{N}}$, respectively) is a permutation over Z_N^* ($Z_{\tilde{N}}^*$ respectively) since $\text{gcd}(N, \phi(N)) = 1$ ($\text{gcd}(\tilde{N}, \phi(\tilde{N})) = 1$ respectively) is trivial if N (\tilde{N} , respectively) is product of two large prime numbers.

The common reference string crs is $(\text{crs}_{\text{DE}}, \text{crs}_{\text{CM}}, \text{ZK-PRoPrimes})$.

The initialization protocol. The initialization protocol enables a sender to set up a verifiably committed database. That is, on input n messages (m_1, \dots, m_n) ($m_i \in Z_{\tilde{N}}$) and the common reference string crs , a sender S performs the following computations

- S invokes the Damgård-Nielsen’s mixed commitment scheme to generate n commitments $c_i = \text{COM}_{K_i}(m_i, r_i) \bmod \tilde{N}^2 = K_i^{m_i} r_i^{\tilde{N}} \bmod \tilde{N}^2$, where $m_i \in Z_{\tilde{N}}$ and $r_i \in Z_{\tilde{N}}^*$.
- S invokes the Damgård-Fujisaki’s protocol [11] and proves that he knows m_i and r_i such that $c_i = \text{COM}_{K_i}(m_i, r_i) \bmod \tilde{N}^2$ ($i = 1, \dots, n$). Let DB-PRoK be a zero-knowledge proof of the knowledge $\langle (m_1, r_1), \dots, (m_n, r_n) \rangle$ such that $c_i = \text{COM}_{K_i}(m_i, r_i) \bmod \tilde{N}^2$ ($i = 1, \dots, n$).
- Let $C = (c_1, \dots, c_n)$ and $D = (C, \text{DB-PRoK})$. S then signs D by means of a secure signature scheme which is independent of the described OT protocol (say the Cramer-Shoup’s signature scheme [9] which is secure in the sense of Goldwasser, Micali and Rivest [12]). Let Σ_D be a signature of D and $\text{DB} = (D, \Sigma_D)$ (we call DB a verifiably committed database). The public database DB is then broadcasted to all participants.

The i th adaptive OT-query protocol. On input common reference strings $\text{crs}_{\text{DE}} = \langle \text{des}(G), (g_1, h_1), \dots, (g_n, h_n), N \rangle$, $\text{crs}_{\text{MC}} = \langle \tilde{N}, K_1, \dots, K_n \rangle$ and the public database $\text{DB} = (D, \Sigma_D)$, the sender S and the receiver R jointly runs the following 3-step communications:

Step 1: R first verifies the validity of public database $\text{DB} = (D, \Sigma_D)$ (i.e., verifying the signature of the database). If it is not valid, then outputs \perp ; If the database DB is valid, R continues the following procedures:

- On input the retrieved $(i - 1)$ messages $m_{\sigma_1}, \dots, m_{\sigma_{i-1}}$ and its random string r_R , R adaptively outputs the i th index $\sigma_i \in [1, n]$;
- on input crs and $\sigma_i \in [1, n]$, R randomly chooses $z_{\sigma_i} \in [0, N^2/4]$, and computes $g = g_{\sigma_i}^{z_{\sigma_i}} \bmod N^2$ and $h = h_{\sigma_i}^{z_{\sigma_i}} \bmod N^2$. Let $\text{tpk} = (g, h)$ and $\text{tsk} = z_{\sigma_i}$. R then invokes the Cramer-Damgård-Schoenmakers’ OR-protocol [8] and proves to the sender that

$$(g_1, h_1, g, h) \vee (g_2, h_2, g, h) \vee \dots \vee (g_n, h_n, g, h)$$

is a Diffie-Hellman quadruple. Let ZK-PRoOR be the zero-knowledge proof of the relationship via the Cramer-Damgård-Schoenmakers’ OR-protocol. R keeps tsk as a secret and sends tpk together ZK-PRoOR to S .

Step 2: On input crs , tpk and ZK-PRoOR, the sender S checks the following three conditions: 1) $g \in Z_{N^2}^*$ and $h \in Z_{N^2}^*$; 2) $g^{2N} \neq 1$ and $h^{2N} \neq 1$ and 3) the transcript ZK-PRoOR is valid. If any of three conditions is violated, then outputs \perp ; otherwise, S performs the following computations:

1. S randomly chooses $s_j, t_j \in [0, N^2/4]$;
2. S computes $u_j = g_j^{s_j} h_j^{t_j} \bmod N^2$, $v_j = g^{s_j} h^{t_j} (1 + N)^{m_j} \bmod N^2$; Let $E_j = (u_j, v_j)$. S then invokes the Damgård-Fujisaki’s protocol [11] and proves that c_j (the ciphertext c_j is generated in the initial stage) and E_j are ciphertexts of the same message m_j . Let ZK-PKoEQ be a zero-knowledge proof of the equality of two ciphertexts E_i and c_i ($i = 1, \dots, n$). S then sends n ciphertexts (E_1, \dots, E_n) together with ZK-PKoEQ to R .

Step 3: Upon receiving (E_1, \dots, E_n) and ZK-PKoEQ, R checks the proof. If the check is valid, R decrypts E_{σ_i} to obtain m_{σ_i} , otherwise, outputs \perp .

This ends the description of the protocol π .

Theorem 1. *The proposed adaptive oblivious transfer protocol $\text{OT}_{k \times 1}^n$ is universally composable in the $\mathcal{F}_{\text{crs}}^D$ -hybrid model in the presence of static adversaries under the joint assumptions that the underlying signature scheme is secure in the sense of Goldwasser, Micali and Rivest [12], the decisional Diffie-Hellman problem over G and the decisional composite residuosity problem in Z_{N^2} are hard as well as all knowledge proof protocols are (either statistical or computational) zero-knowledge against malicious participants.*

Proof. Let \mathcal{A} be a static adversary that interacts with the parties S and R running the protocol π . We will construct an ideal world adversary \mathcal{S} interacting with the ideal functionality $\text{OT}_{k \times 1}^n$ such that no environment \mathcal{Z} can distinguish an interaction with \mathcal{A} in the protocol π from an interaction with the simulator \mathcal{S} in the ideal world.

Simulating the corrupted sender. When S is corrupted and R is honest, the adversary \mathcal{A} gets S ’s input from the environment \mathcal{Z} , and generates all the messages from S . The goal of a simulator \mathcal{S} now is to generate the remaining messages (namely, messages from R) so that the entire transcript is indistinguishable from the real interaction between S and R from the point view of the environment \mathcal{Z} . The details of simulator for corrupted party S is described below:

1. when the environment \mathcal{Z} queries to $\text{OTcrsGen}(1^\kappa)$ for a common reference string crs , the simulator invokes the key generation algorithm \mathcal{G} of the BCP’s encryption to generate composite modulus of the form $N = pq$ that is a product of two safe primes p and q (i.e., $p = 2p' + 1$, $q = 2q' + 1$), a cyclic group $G \subseteq Z_{N^2}^*$ of order N' ($N' = p'q'$), and n random generators g_1, \dots, g_n of G and n random elements (h_1, \dots, h_n) such that $h_i = g_i^{x_i} \bmod N^2$, $x_i \in_U Z_{N'}$, $i = 1, \dots, n$. The simulator keeps the auxiliary strings (p, q) and (x_1, \dots, x_n) secret. Let $\text{crs}_{\text{DE}} = \langle \text{des}(G), (g_1, h_1), \dots, (g_n, h_n), N \rangle$.

OTcrsGen then invokes the key generation algorithm of Damgård-Nielsen’s mixed commitment scheme to generate a public key \tilde{N} and n extractable keys $K_1, \dots, K_n \in_U Z_{N^2}^*$, where $\tilde{N} = \tilde{p}\tilde{q}$, $\tilde{p} = 2\tilde{p}' + 1$, $\tilde{q} = 2\tilde{q}' + 1$, and \tilde{p} , \tilde{p}' , \tilde{q} and \tilde{q}' are large prime numbers.

Let $K_i = (1 + \tilde{N})^{k_i} r_{k_i}^{\tilde{N}} \bmod \tilde{N}$ ($i = 1, \dots, n$). The simulator keeps (\tilde{p}, \tilde{q}) and $\langle (k_1, r_{k_1}), \dots, (k_n, r_{k_n}) \rangle$ secret. Let $\text{crs}_{\text{MC}} = \langle \tilde{N}, K_1, \dots, K_n \rangle$.

Given (p, q) and (\tilde{p}, \tilde{q}) such that $N = pq$ and $\tilde{N} = \tilde{p}\tilde{q}$, the simulator further invokes Camenisch and Michels protocol and proves in zero-knowledge that N (\tilde{N} , respectively) is a product of two safe primes p and q (\tilde{p} and \tilde{q} respectively). The transcript of zero-knowledge proof is denoted by ZK-PRoPrimes . Let $\text{crs} = (\text{crs}_{\text{DE}}, \text{crs}_{\text{MC}}, \text{ZK-PRoPrimes})$. $\mathcal{F}_{\text{crs}}^{\mathcal{D}}$ returns (sid, crs) to the environment \mathcal{Z} ;

2. when the simulator \mathcal{S} receives $(\text{sid}, \text{ssid}_i, (D, \Sigma_D))$ from the real world adversary \mathcal{A} who fully controls the corrupted sender S . The simulator checks the validity of DB. If the signature is invalid, \mathcal{S} outputs \perp .
3. If DB is valid, then the simulator \mathcal{S} extracts the messages (m_1, \dots, m_n) from the given ciphertexts (c_1, \dots, c_n) using the trapdoor strings (\tilde{p}, \tilde{q}) and $\langle (k_1, r_{k_1}), \dots, (k_n, r_{k_n}) \rangle$ and then forwards the extracted message m_1, \dots, m_n to the ideal functionality $\mathcal{F}_{\text{OT}_{k \times 1}^n}$. We stress that the simulator \mathcal{S} must send the messages (m_1, \dots, m_n) to the functionality $\mathcal{F}_{\text{OT}_{k \times 1}^n}$ at the initial stage.
4. \mathcal{S} randomly selects (g, h) with order N' and sets $pk = (g, h)$. We stress that the choice of (g, h) is a trivial task since \mathcal{S} holds the master key (p, q) . \mathcal{S} invokes Cramer-Damgård-Schoenmakers' OR-protocol to prove that there exists an index $i^* \in [1, n]$ such that (g_{i^*}, h_{i^*}, g, h) is a Diffie-Hellman quadruple. Let ZK-PRoOR a transcript of OR-protocol. The simulator then sends $tpk = (g, h)$ and ZK-PRoOR to the adversary \mathcal{A} .
5. Upon receiving (E_1, \dots, E_n) and ZK-PKoEQ from the adversary \mathcal{A} , the simulator \mathcal{S} checks the given proof. If the check is valid, \mathcal{S} decrypts E_{σ_i} to reveal m_{σ_i} , otherwise, outputs \perp .

Let $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ be the view of ideal world adversary \mathcal{S} described above and $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}$ be the view of real world adversary \mathcal{A} of protocol π . Notice that when \mathcal{S} receives $(\text{sid}, \text{ssid}_i, (D, \Sigma_D))$ from the real world adversary \mathcal{A} , it checks the validity of DB. If the signature is invalid, \mathcal{S} outputs \perp . If the signature is valid, \mathcal{S} extracts the messages (m_1, \dots, m_n) from the given ciphertexts (c_1, \dots, c_n) using the trapdoor string (\tilde{p}, \tilde{q}) to the ideal functionality $\mathcal{F}_{\text{OT}_{k \times 1}^n}$. This means that the rewinding technique for extract implicit messages of the corrupted sender is not applied here. The only difference between $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ is the different strategies to generate the public key (g, h) in π and the public key (g, h) generated in the simulation stage. By the Diffie-Hellman assumption over $Z_{N^2}^*$, we know that $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$.

Simulating the corrupted receiver. When S is honest and R gets corrupted, the adversary \mathcal{A} gets R 's input from the environment \mathcal{Z} , and generates all the messages from R . The goal of the simulator now is to generate the remaining messages (namely, all messages from S) so that the entire transcript is indistinguishable from the real interaction between S and R from the point view of the environment \mathcal{Z} .

1. when the environment \mathcal{Z} queries to $\text{OTcrsGen}(1^\kappa)$ for a common reference string crs , the simulator \mathcal{S} invokes a key generation algorithm \mathcal{G} of the BCP's

key generation algorithm to generate crs_{DE} , where $\text{crs}_{\text{DE}} = \langle \text{des}(G), (g_1, h_1), \dots, (g_n, h_n), N \rangle$. The trapdoor string τ is (x_1, \dots, x_n) . The simulator is given the master auxiliary string $\langle p, q \rangle$.

The simulator then invokes the key generation crs_{MC} to generate $\text{crs}_{\text{MC}} = \langle \tilde{N}, K_1, \dots, K_n \rangle$, where $K_j = \psi(0, k_j)$ (i.e., all K_j are E-keys). The auxiliary string is (k_1, \dots, k_n) . The simulator keeps the auxiliary information \tilde{p} and \tilde{q} such that $\tilde{N} = \tilde{p}\tilde{q}$, $\tilde{p} = 2\tilde{p}' + 1$, $\tilde{q} = 2\tilde{q}' + 1$ secret.

Let $\text{crs} = (\text{crs}_{\text{DE}}, \text{crs}_{\text{MC}})$. $\mathcal{F}_{\text{crs}}^{\mathcal{D}}$ returns (sid, crs) to the environment \mathcal{Z} ;

2. the simulator \mathcal{S} invokes the Damgård-Nielsen’s mixed commitment scheme to generate n commitments $C_i = \text{COM}_{K_i}(m_i, r_i) \bmod \tilde{N}^2 = K_i^{m_i} r_i^{\tilde{N}} \bmod \tilde{N}^2$, where $m_i = 0$ (all are dummy messages) and $r_i \in Z_{\tilde{N}}^*$.
3. Given crs , tpk and ZK-PRoOR, the simulator \mathcal{S} checks the following three conditions: 1) checking $g \in Z_{N^2}^*$ and $h \in Z_{N^2}^*$; 2) checking $g^{2N} \neq 1$ and $h^{2N} \neq 1$ and 3) checking the validity of the transcript ZK-PRoOR. If any of three conditions is violated, then outputs \perp ; otherwise, \mathcal{S} extracts an index σ_i by testing the equation $h \stackrel{?}{=} g^{x_{\sigma_i}}$ ($i = 1, \dots, n$). \mathcal{S} sends σ_i to the ideal functionality $\mathcal{F}_{\text{OT}_{k \times 1}^n}$ and obtains m_{σ_i} .
4. \mathcal{S} modifies the internal states to generate a transcript that is consistent with the given database DB according to the following strategy: given c_{σ_i} (the encryption of dummy message with randomness r_{σ_i}), the simulator extracts a new randomness r'_{σ_i} from the equation $K_{\sigma_i}^0 r_{\sigma_i}^{\tilde{N}} = K_{\sigma_i}^{m_{\sigma_i}} r'_{\sigma_i}{}^{\tilde{N}}$, where $K_{\sigma_i} = \psi(0, k_{\sigma_i})$.
The simulator then invokes the Damgård-Fujisaki’s protocol to output a transcript that c_j (the ciphertext c_j is generated in the initial stage) and E_j are ciphertexts of the same message m_j . Let ZK-PKoEQ be a transcript of the proof. \mathcal{S} then sends n ciphertexts (E_1, \dots, E_n) together with ZK-PKoEQ to the adversary \mathcal{A} .

Let $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ be the view of ideal world adversary \mathcal{S} described above and $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}$ be the view of real world adversary \mathcal{A} of protocol π . Notice that \mathcal{S} extracts an index σ_i by testing the equation $h \stackrel{?}{=} g^{x_{\sigma_i}}$ ($i = 1, \dots, n$) and then sends σ_i to the ideal functionality $\mathcal{F}_{\text{OT}_{k \times 1}^n}$ to learn m_{σ_i} . This means that the implicit input of the corrupted receiver is extracted by the auxiliary string (x_1, \dots, x_n) . The only difference between $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ is the generation of n keys (K_1, \dots, K_n) of the Damgård-Nielsen’s mixed commitment. By the key indistinguishability assumption, we know that $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$.

A close look at the proof shows that both statistical zero-knowledge proof protocols and computational zero-knowledge proof protocols work in our scheme.

Combining the above statements, we know that the proposed adaptive oblivious transfer protocol $\text{OT}_{k \times 1}^n$ is universally composable in the $\mathcal{F}_{\text{crs}}^{\mathcal{D}}$ -hybrid model assuming that the decisional Diffie-Hellman problem over G is hard as well as all knowledge proof protocols are zero-knowledge. \square

4 Conclusion

In this paper, a new implementation of adaptive oblivious transfer protocol $OT_{k \times 1}^n$ has been proposed. We have shown that the proposed scheme has achieved the UC-security in the $\mathcal{F}_{\text{crs}}^{\mathcal{D}}$ -hybrid model under the joint assumptions that the underlying signature scheme is secure, the decisional Diffie-Hellman problem and the decisional composite residuosity problem in Z_{N^2} are hard as well as all knowledge proof protocols applied in this paper are computational zero-knowledge in the presence of static adversaries.

References

1. Bresson, E., Catalano, D., Pointcheval, D.: A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003)
2. Camenisch, J., Neven, G., Shelat, A.: Simulatable Adaptive Oblivious Transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
3. Camenisch, J., Michels, M.: Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
4. Canetti, R.: A new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)
5. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
6. Canetti, R.: Obtaining Universally Composable Security: Towards the Bare Bones of Trust. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 88–112. Springer, Heidelberg (2007)
7. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC 2002, pp. 494–503 (2002)
8. Cramer, R., Damgård, I., Schoenmakers, B.: Proof of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
9. Cramer, R., Shoup, V.: Signature scheme based on the Strong RAS assumption. In: 6th ACM Conference on Computer and Communication Security, Singapore, November 1999. ACM Press, New York (1999)
10. Damgård, I., Nielsen, J.B.: Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
11. Damgård, I., Fujisaki, E.: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
12. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2), 281–308 (1988)
13. Green, M., Hohenberger, S.: Blind Identity-Based Encryption and Simulatable Oblivious Transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)

14. Green, M., Hohenberger, S.: Universally composable adaptive oblivious Transfer. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 179–197. Springer, Heidelberg (2008)
15. Kurosara, K., Nojima, R.: Simple adaptive oblivious transfer without random oracle. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912. Springer, Heidelberg (2009) (available at eprint.iacr.org)
16. Naor, M., Pinkas, B.: Oblivious Transfer with Adaptive Queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
17. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
18. Peikert, C., Vaikuntanathan, V., Waters, B.: A Framework for Efficient and Composable Oblivious Transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
19. Zhu, H.: Round Optimal Universally Composable Oblivious Transfer Protocols. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 328–334. Springer, Heidelberg (2008)
20. Zhu, H.: New Constructions for Reusable, Non-erasure and Universally Composable Commitments. In: ISPEC 2009, pp. 102–111 (2009)
21. Zhu, H., Bao, F.: Constructing Universally Composable Oblivious Transfers from Double Trap-Door Encryptions. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 323–333. Springer, Heidelberg (2009)

Discrete-Log-Based Additively Homomorphic Encryption and Secure WSN Data Aggregation

Licheng Wang^{1,2}, Lihua Wang², Yun Pan³, Zonghua Zhang², and Yixian Yang¹

¹ Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications
10 West Tucheng Road, Beijing, P.R. China 100876

² Security Fundamental Group, Information Security Research Center
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi, Koganei-shi, Tokyo, 184-8795 Japan

³ School of Computer, Communication University of China
1 East Street of Dingfuzhuang, Beijing, P.R. China 100024
{wanglc, wlh, zonghua}@nict.go.jp, pany@cuc.edu.cn,
{wanglc, yxy}@bupt.edu.cn

Abstract. At PKC 2006, Chevallier-Mames, Paillier, and Pointcheval proposed encryption schemes that are partially homomorphic, either additively or multiplicatively and announced an open research problem: finding a discrete-log-based cryptosystem that would help realize fully additive or multiplicative homomorphism. In this study, we achieve this goal by lifting the message space of the ElGamal scheme from \mathcal{M} to $g_0^{\mathcal{M}}$. We then apply our scheme for constructing a novel protocol for secure data aggregation in Wireless Sensor Networks.

Keywords: Discrete-logarithm problem, additively homomorphic encryption, wireless sensor networks, data aggregation.

1 Introduction

1.1 Background: Homomorphic Encryption

In general, we expect a cryptosystem to be as secure as possible. To this end, various security notions have been developed. The basic requirement for a cryptosystem is that adversaries must be prevented from learning confidential messages. This is the so-called security notion of one-wayness (OW) and was recognized even before 2500 B.C. With the development of modern cryptography, particularly after the advent of public-key cryptosystems [7], new desirable security notions were conceived. Naor suggested that different security notions for encryption should be defined by orthogonally considering the various possible goals and the various possible attack models [1]. Typically, two goals, namely, indistinguishability (IND) [13] and non-malleability (NM) [8], and three attack models, namely, chosen-plaintext attack (CPA), non-adaptive chosen-ciphertext attack (CCA1) [14], and adaptive chosen-ciphertext attack (CCA2) [16], have

been considered. Currently, IND-CCA, especially IND-CCA2, is known to be the most desirable security property for many applied cryptosystems.

However, people also visualized a lot of scenarios in which IND-CCA is too rigid to be flexible. It might be possible to perform some positive transformations on a ciphertext without decryption. Thus, the idea of homomorphic encryption was proposed. Homomorphic encryption enables blind transformations on plaintexts via (possibly different) algebraic operations on ciphertexts [6,4]. Depending upon the specific viewpoint, this is either a positive or a negative attribute of a cryptosystem. Homomorphic encryption schemes are inherently malleable and are thus unsuitable in scenarios in which modifications to ciphertexts are forbidden. On the other hand, the homomorphic property is useful for building secure voting systems, collision-resistant hash functions, private information retrieval schemes, etc. At PKC 2006, Chevallier-Mames, Paillier, and Pointcheval [4] proposed encryption schemes that are partially homomorphic, either additively or multiplicatively and finally announced an open research problem: finding a discrete-log-based cryptosystem that helps realize fully additive or multiplicative homomorphism. Note that there are many additively homomorphic encryption schemes based on factoring-related assumptions, such as Goldwasser-Micali's scheme [12], Benaloh's scheme [2], and Paillier's scheme [15]. Most recently, Gentry proposed a lattice-based fully homomorphic encryption scheme [11].

1.2 Background II: Secure WSN Data Aggregation

A very typical application of homomorphic encryption in practice is data aggregation in wireless sensor networks (WSN) [19,5,20,3]. Aggregation techniques are used to reduce the amount of data communicated within a WSN so as to conserve battery power. Since measurements are recorded by individual sensors, they need to be periodically collected and processed to yield data representative of the entire WSN, such as the average and/or variance of the temperature or humidity within an area [3]. One natural approach to data aggregation is to simply add up values as they are forwarded toward the sink. Homomorphic encryption allows some types of statistical computations on encrypted data, similar to those on plaintext data [10], thereby offering significant advantages in securing WSN data aggregation by preventing eavesdropping attacks. If homomorphic encryption is not used, the intermediate sensors must have access to the secret keys in order to decrypt the collected data before they are added. However, we know that methods in which sensitive information such as secret keys is stored inside sensors are prone to many attacks [3]. Once the secret keys are exposed, the network is no longer secure. Homomorphic encryption prevents this problem since sensitive information is not stored on the intermediate sensors [3].

While it is intuitive to use homomorphic encryption for secure data aggregation in WSN, no silver bullet application has been seen so far, and the majority of available schemes are vulnerable to some extent. For example, in [3], an additively homomorphic encryption for WSN data aggregation was proposed. The scheme is essentially a stream cipher and its homomorphic property relies on the synchronization among the key-stream generators, i.e., all sensors in the

field must share the same key-stream generator. Clearly, this is impractical in a distributed environment. More seriously, this approach has two fundamental flaws: (1) it is difficult to ensure the confidentiality of this commonly shared key-stream generator and (2) the scheme would be insecure if some sensors that have access to the key-stream generator were colluded by the adversaries. Therefore, to achieve secure WSN data aggregation, we prefer homomorphic public-key encryption (Hom-PKE) than secret-key encryption. In public-key encryption, no secret is embedded in the sensors, and thus, this scheme is immune to colluding attacks. Vaidehi [10] tried to solve this problem by using a privacy homomorphism called DF-a-New-PH. However, we found that DF-a-New-PH does not have the desired effect of encryption for two reasons: (1) some secret parameters can be easily computed from the public parameters and (2) all secret parameters are explicitly involved in the encryption algorithm. We know that no public-key encryption scheme can operate in this manner. DF-a-New-PH is essentially an encoding/decoding system rather than a cryptosystem since it does not provide confidentiality. In DF-a-New-PH, all secrets can be made public! Of course, when using Hom-PKE, a malicious node can still insert junk data to disturb the aggregation. However, this problem is not specific to Hom-PKE. All encryption schemes, public-key or secret-key, are vulnerable to this kind of attack. In terms of consequence, this kind of attack can be viewed as denial of service (DOS) attack. How to resist DOS or DDOS attacks is another significant research topic that is beyond the scope of this paper.

1.3 Motivation and Contributions

In this paper, we first try to answer the open problem raised by Chevallier-Mames, Paillier, and Pointcheval [4]. We achieve this goal by lifting the message space in the ElGamal scheme. This yields a discrete-logarithm-based additively homomorphic encryption scheme. Then, we design a new protocol for WSN data aggregation by employing the proposed additively homomorphic encryption scheme.

2 Intractability of Discrete Logarithm Problems

Definition 1 (Discrete Logarithm Problem, DLP). *Let G be a finite cyclic group of order n and g be a generator. For any $h \in G$, the discrete logarithm of h with respect to g , denoted $DLOG_g(h)$, is the element $x \in \mathbb{Z}_n$ such that $h = g^x$.*

In particular, in the remain sections we assume that $G = \mathbb{Z}_p^*$, where p is a large prime. The security of the Diffie-Hellman protocol [7], as well as ElGamal encryption scheme [9] rely on the intractability of DLP over \mathbb{Z}_p^* for secure prime p . Note that not all discrete logarithm problems are intractable. In fact, the difficulty of computing discrete logarithms in \mathbb{Z}_p^* is determined by the size of the largest prime factor of $p - 1$. More specifically, given the prime factoring $p - 1 = \prod_{i=1}^r q_i^{e_i}$ and $q = \max\{q_1, \dots, q_r\}$, the running time for computing x is bounded by $q^{1/2}len(p)^{O(1)}$ [18], where $len(\cdot)$ indicates the bit-length of a given number. This suggests that when $p - 1$ does not contain any large prime factor,

the discrete logarithm problem over \mathbb{Z}_p^* can be solved efficiently. A prime p is called *secure prime* if $p - 1$ contains large, say at least 160 bits, prime factors.

In subsequent contexts, we assume that $q, p = 2q + 1$ are large primes and $G = \langle g \rangle \subset \mathbb{Z}_p^*$ be a cyclic group of order q generated by g . Other frequently used DLOG-based cryptographic problems are

Definition 2 (Computational Diffie-Hellman problem, CDH). *Given $g^x, g^y \in G$ for some unknown $x, y \in \mathbb{Z}_q$, compute $g^{xy} \in G$.*

Definition 3 (Decisional Diffie-Hellman problem, DDH). *Given two distributions $D = (g^x, g^y, g^{xy}) \in G^3$ and $R = (g^x, g^y, g^z) \in G^3$ for randomly distributed $x, y, z \in \mathbb{Z}_q$, distinguish D from R .*

It is easily seen that $DDH \preceq_P CDH \preceq_P DLP$ where \preceq_P denotes polynomial reduction between computational problems. At present, we know that for a secure prime p , DLP, CDH, and DDH are intractable, except for resorting a quantum computer [17].

3 The Proposed Scheme

3.1 OW-CPA ElGamal and Multiplicative Homomorphism

Recall that the original version of the ElGamal cryptosystem [9] is multiplicatively homomorphic, and thus, let us denote it by $\times\text{HomElG}$. The encryption algorithm in $\times\text{HomElG}$ is denoted by \mathcal{E}_\times and is multiplicatively homomorphic in the sense that under the same public key $y = g^x \bmod p$ (where g is a generator of some q order subgroup of \mathbb{Z}_p^* with $q|p - 1$, both p and q are large primes and x is the private key picked randomly from \mathbb{Z}_q), given two ciphertexts

$$\mathcal{E}_\times(y; m_1) = (g^{r_1} \bmod p, y^{r_1} m_1 \bmod p), \text{ and} \tag{1}$$

$$\mathcal{E}_\times(y; m_2) = (g^{r_2} \bmod p, y^{r_2} m_2 \bmod p), \tag{2}$$

anyone can, without knowing the secret key x , the original messages m_i , and the random numbers r_i ($i = 1, 2$), compute the ciphertext for the plaintext $m_1 m_2$ by using the following formula:

$$\mathcal{E}_\times(y; m_1 m_2) = (g^{r_1} \cdot g^{r_2} \bmod p, y^{r_1} m_1 \cdot y^{r_2} m_2 \bmod p). \tag{3}$$

More formally, given two valid ciphertexts (c_{11}, c_{12}) and (c_{21}, c_{22}) for the messages m_1 and m_2 under a common public key, $(c_{11}, c_{12}) \odot (c_{21}, c_{22})$ would be a valid ciphertext for the message $m_1 \cdot m_2$ under the same public key, where \odot denotes component-wise modular multiplication, i.e.,

$$(c_{11}, c_{12}) \odot (c_{21}, c_{22}) = (c_{11} \cdot c_{21} \bmod p, c_{12} \cdot c_{22} \bmod p). \tag{4}$$

Note that this version of ElGamal merely achieves OW-CPA security, i.e., one-wayness against chosen plaintext attacks.

3.2 Message Lifting and Parameter Selection

A natural idea for converting a multiplicatively homomorphic scheme into an additively homomorphic one is to lift the message space \mathcal{M} to the exponential space $g_0^{\mathcal{M}}$ for some base g_0 . To maintain the compactness of the lifted message space, we choose a prime p_0 such that $p_0 - 1$ does not contain large prime factors and then define the following *lifting* operation:

$$m \mapsto g_0^m \bmod p_0, \forall m \in \{0, \dots, p_0 - 2\}, \tag{5}$$

where g_0 is a primitive root of $\mathbb{Z}_{p_0}^*$. Further, let $L_{g_0}(\cdot)$ indicate the reverse, i.e., an *unlift* operation that is defined by

$$L_{g_0}(g_0^m \bmod p_0) = m \bmod (p_0 - 1). \tag{6}$$

Note that the precision is determined by the size of the encoding space. Thus, the precision increases with the prime p_0 . Typically, we can choose $p_0 = 2^{16} + 1$ and $g_0 = 3$. Then, the sensed data are 16 bits long and the sensing precision is $\frac{d_{max} - d_{min}}{2^{16}}$, where d_{max} and d_{min} are potential maximum and minimum of the concerned data. This is perhaps suitable for most WSN-oriented applications in which the relevant data would be temperature, humidity, atmospheric pressure, noise, etc.

3.3 DLP-Based Additive Homomorphism

Now, let us proceed to develop an additively homomorphic ElGamal encryption scheme, denoted $^+ \text{HomElG}$. This scheme is based on the original multiplicatively homomorphic ElGamal scheme, $^{\times} \text{HomElG}$.

- Key-generation algorithm $\mathcal{K}(1^k)$: Take as input the security parameter k , choose two primes p and p_0 such that
 - (1) $p = 2q + 1$ for some large prime q ,
 - (2) $p_0 = 2t^\kappa + 1 < p$ for a small prime t and some positive integer κ .

And then, select two generators g and g_0 so that g generates a q -order subgroup of \mathbb{Z}_p^* , while g_0 generates the group $\mathbb{Z}_{p_0}^*$. (Typically, let $t = 2, \kappa = 15$ and $g_0 = 3$.) Next, randomly select a private key x from \mathbb{Z}_q and compute the corresponding public key $y = g^x \bmod p$. Finally, output the system public parameters

$$params = (p, q, g; p_0, g_0) \tag{7}$$

and return the public/private key pair (y, x) . Note that the message space is $\mathcal{M} = \{0, \dots, p_0 - 2\} \subset \mathbb{Z}_{p_0}^*$, while the ciphertext space is $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_{p_0}^*$. The security of the scheme is rooted in the secure prime p and irrelevant to p_0 .

Remark 1. Under this condition, we have that: (1) $|\mathbb{Z}_{p_0}^*| = p_0 - 1$ and thus $(g_0^m \bmod p_0)$ goes through the set $\{1, \dots, p_0 - 1\}$ when m goes through the set $\{0, \dots, p_0 - 2\}$; and (2) the maximal prime factor of $(p_0 - 1)$ is t , and thus the unlifting operation can be finished efficiently. Note that our scheme also works for other form prime p_0 when $p_0 - 1$ does not contain large prime

factors. But for simplicity, we merely select p_0 taking the form $p_0 = 2t^\kappa + 1$ for some small prime t and some positive integer κ . Then, the time complexity of unlifting operation is bounded by $\sqrt{2} \log^c p_0$ for some constant c [18].

- Encryption algorithm $\mathcal{E}_+(y; m)$: Take as inputs the public key y and a message $m \in \mathcal{M}$, choose $r \in \mathbb{Z}_q$ at random and output the ciphertext (c_1, c_2) , where

$$c_1 = g^r \text{ mod } p, \quad c_2 = y^r \cdot (g_0^m \text{ mod } p_0) \text{ mod } p. \tag{8}$$

- Decryption algorithm $\mathcal{D}_+(x; (c_1, c_2))$: Take as inputs the secret key x and a ciphertext (c_1, c_2) , compute the plaintext m by using the decryption algorithm of $\times\text{HomElG}$ and then perform unlifting, i.e.,

$$m = L_{g_0}(\mathcal{D}_\times(x; (c_1, c_2))), \tag{9}$$

where \mathcal{D}_\times is defined by

$$\mathcal{D}_\times(x; (c_1, c_2)) = c_2 / c_1^x \text{ mod } p. \tag{10}$$

Remark 2. In practice, we need pay attention to the issue of overflow¹. Notice that when $(g_0^m \text{ mod } p_0)^N > p$, the following inequality

$$((g_0^m \text{ mod } p_0)^N \text{ mod } p) \text{ mod } p_0 \neq ((g_0^{m \cdot N} \text{ mod } p_0 \text{ mod } p) \text{ mod } p) \text{ mod } p_0 \tag{11}$$

might hold. This suggests the ciphertext on the message $N \cdot m$ cannot be correctly formed by using the proposed scheme. Considering there exists m^* such that $g_0^{m^*} \text{ mod } p_0$ achieves the maximum, $p_0 - 1$. In order to compute the ciphertext of $N \cdot m^*$ correctly by using the above additive homomorphism, N should satisfy $N < \log p / \log(p_0 - 1)$, where N indicates the maximal number of the messages than can be correctly added up. This undesirable restrictions might bring inconvenient for certain applications. For example, if p and p_0 are 1024-bit and 16-bit primes as aforementioned, $N \leq 64$.

The following theorem that captures the consistency and the security of the proposed scheme.

Theorem 1. *Under the same public key y and for $\forall m_1, m_2 \in \mathcal{M}$, we have that*

$$\mathcal{E}_+(y; m_1) \odot \mathcal{E}_+(y; m_2) = \mathcal{E}_+(y; m_1 + m_2), \tag{12}$$

and under the same private key x and for $\forall (c_{11}, c_{12}), (c_{21}, c_{22}) \in \mathcal{C}$, we have that

$$\mathcal{D}_+(x; (c_{11}, c_{12})) + \mathcal{D}_+(x; (c_{21}, c_{22})) \equiv \mathcal{D}_+(x; (c_{11}, c_{12}) \odot (c_{21}, c_{22})) \pmod{p_0 - 1}. \tag{13}$$

Proof. See Appendix A.

Theorem 2. *The proposed cryptosystem is indistinguishable against chosen plaintext attack assuming that the decisional Diffie-Hellman problem over \mathbb{Z}_p^* (where p is a secure prime) is intractable.*

Proof. Omitted for space limitation.

¹ This issue is noticed by an anonymous referee (cf. Acknowledgments).

4 Protocol for Secure WSN Data Aggregation

Without loss of generality, let us visualize a multilevel network tree in which numerous sensor nodes and only one sink node exist. We assume that all nodes are potential aggregators and that data get aggregated as they propagate toward the sink. Suppose the system public parameters, as well as the sink's public key y , are input to each sensor node beforehand. Now, the protocol for secure WSN data aggregation involves the following steps:

- *Parameters selection.* Suppose that the maximum number of potential sensor nodes is at most N ² the range of the sensed data is $[d_{min}, d_{max}]$, and the desired sensing precision is δ . Then, we should select prime p_0 so that $p_0 - 1 > N \frac{d_{max} - d_{min}}{\delta}$. Finally, let us select a generator g_0 for the group $\mathbb{Z}_{p_0}^*$. (E.g., in sensing the temperature of the atmosphere, it is enough to set $d_{min} = -50$, $d_{max} = 50$ and $\delta = 0.01$. Now, suppose that the number of involved sensors is no more than $N = 20000$. Then, we can set $t = 3, \kappa = 17$ and then $p_0 = 2 \cdot 3^{17} = 258280327$ and $g_0 = 5$.)
- *Encryption.* Each leaf sensor node s_i encrypts the corresponding data m_i by using the algorithm $c^{(i)} = \mathcal{E}_+(y; m_i)$ and then sends $c^{(i)}$ to its parent sensor node.
- *Aggregation.* Each intermediate sensor node, s_j , collects all ciphertexts from its children nodes, s_{j_1}, \dots, s_{j_r} w.l.o.g., and then aggregates them by

$$c^{(j)} = c^{(j_1)} \odot c^{(j_2)} \odot \dots \odot c^{(j_r)} \odot \mathcal{E}_+(y; m_j), \quad (14)$$

where m_j is the value measured by s_j itself. Then, s_j transmits $c^{(j)}$ to its parent sensor node, where similar aggregation is performed.

- *Decryption.* Finally, suppose that the sink node, say s_0 , collects all ciphertexts from its children sensor nodes.
 - (1) At first, s_0 performs the aggregation operation in an identical manner to the general intermediate sensor nodes and obtains the ciphertext $c^{(0)} = (c_1^{(0)}, c_2^{(0)})$.
 - (2) Then, s_0 extracts the aggregated data m by performing the decryption

$$m = \mathcal{D}_+(x; c^{(0)}). \quad (15)$$

Clearly, $m = \sum_i m_i$ holds. (Note that, since $N/\delta < p_0 - 1$, thus the formula (13) holds exactly even without modular operation. This condition is crucial for ensuring the correctness of summation obtained by using our additively homomorphic encryption scheme ⁺HomElG.)

Additive aggregation can also be used to compute variance, standard deviation, and any other moments of the measured data [CMT05] even when the difference between the value obtained at each sensor node and the average of all values

² We can make a rough estimation on N since our protocol is not sensitive to this boundary.

obtained at the sensor nodes is unknown. Of course, for this purpose, the above aggregation protocol should be updated accordingly. For example, when computing the variance, each leaf sensor node computes and transmits $\mathcal{E}_+(y; m_i)$ as well as $\mathcal{E}_+(y; m_i^2)$ (i.e., the ciphertext of the square of m_i) to its parent sensor node, which in turn aggregates not only $\widehat{S}^{(1)} = \bigodot_i \mathcal{E}_+(y; m_i) = \mathcal{E}_+(y; \sum_{i=1}^n m_i)$ but also $\widehat{S}^{(2)} = \bigodot_i \mathcal{E}_+(y; m_i^2) = \mathcal{E}_+(y; \sum_{i=1}^n m_i^2)$. Eventually, the sink obtains

$$DM = EM^2 - (EM)^2 = 1/n \cdot \mathcal{D}_+ \left(x; \widehat{S}^{(2)} \right) - \left(1/n \cdot \mathcal{D}_+ \left(x; \widehat{S}^{(1)} \right) \right)^2, \quad (16)$$

where DM and EM indicate the variance and the expectation of the random variable M defined over the message space \mathcal{M} , which should be re-defined as

$$\mathcal{M} = \left\{ 0, \dots, \left\lfloor \sqrt{(p_0 - 2)/N} \right\rfloor \right\} \quad (17)$$

so as to let the potential maximal value of $\sum_{i=1}^n m_i^2$ lie in the set of $\{0, \dots, p_0 - 2\}$. This also means we should select larger p_0 if it is not convenient to compress \mathcal{M} . Similarly, for computing $k (> 2)$ -order moments, even large primes p_0 should be used. Here, n indicates the total number of sensor nodes involved in measuring data. We assume that intermediate sensor nodes also measure the relevant statistical data. Otherwise, n should be the total number of leaf nodes. A general approach for calculating the appropriate n is to view n as a new statistical variable and count it during the process of counting $\widehat{S}^{(1)}$ and $\widehat{S}^{(2)}$. We can even maintain the anonymity of the sensor nodes involved in the measurement by employing the above protocol for the statistical variable n , i.e., by counting n secretly. This task is essentially equivalent to e-voting, which can be implemented easily by an additively homomorphic encryption scheme.

In addition, by adopting the idea of Yokoo and Suzuki [21], it is not difficult to find the maximum of some secret numbers by using the proposed scheme.

Acknowledgements

We thank the anonymous referees for very useful comments. This work is supported by Japan NICT International Exchange Program (No. 2009-002), China National Natural Science Foundation Programs (No.90718001, No. 60973159), China 973 Program (No. 2007CB310704) and 863 Program (No. 2009AA012439), and Communicate University of China 211-project.

References

1. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
2. Benaloh, J.: Dense probabilistic encryption. In: First Annual Workshop on Selected Areas in Cryptography, Kingston, ON, May 1994, pp. 120–128 (1994)

3. Castelluccia, C., Mykletun, E., Tsudik, G.: Efficient aggregation of encrypted data in wireless sensor networks. In: *MobiQuitous*, pp. 109–117. IEEE Computer Society, Los Alamitos (2005)
4. Chevallier-Mames, B., Paillier, P., Pointcheval, D.: Encoding-free elgamal encryption without random oracles. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) *PKC 2006*. LNCS, vol. 3958, pp. 91–104. Springer, Heidelberg (2006)
5. Damgard, I., Geisler, M., Kroigard, M.: Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography (IJACT)* 1(1), 22–31 (2008)
6. Hoogh, S.D., Schoenmakers, B., Skoric, B., Villegas, J.: Verifiable rotation of homomorphic encryptions. In: Jarecki, S., Tsudik, G. (eds.) *PKC 2009*. LNCS, vol. 5443, pp. 393–410. Springer, Heidelberg (2009)
7. Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Transactions on Information Theory* 22(5), 644–654 (1976)
8. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pp. 542–552. ACM Press, New Orleans (1991)
9. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory (TIT)* 31(4), 469–472 (1985)
10. Ertaul, L., Kedlaya, V.: Computing aggregation function minimum/maximum using homomorphic encryption schemes in wireless sensor networks (WSNs). In: Arabnia, H.R., Clincy, V.A., Yang, L.T. (eds.) *ICWN 2007*, pp. 186–192. CSREA Press (2007)
11. Gentry, C.: On homomorphic encryption over circuits of arbitrary depth. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC 2009)*, pp. 169–178. ACM Press, Stanford University (2009)
12. Goldwasser, S., Micali, S.: Probabilistic encryption. *J. Comput. Syst. Sci. (JCSS)* 28(2), 270–299 (1984)
13. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC 1982)*, pp. 365–377. ACM Press, San Francisco (1982)
14. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen cypher-text attack. In: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC 1990)*, pp. 427–437. ACM Press, Baltimore (1990)
15. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) *EUROCRYPT 1999*. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
16. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) *CRYPTO 1991*. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
17. Shor, P.: Polynomail-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* 5, 1484–1509 (1997)
18. Shoup, V.: *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, Cambridge (2005)
19. Ugus, O., Westhoff, D., Laue, R., Shoufan, A., Huss, S.A.: Optimized implementation of elliptic curve based additive homomorphic encryption for wireless sensor networks, CoRR abs/0903.3900 (2009)
20. Yacoub, H., Sarkar, T.K.: A homomorphic approach for through-wall sensing. *IEEE Trans. Geoscience and Remote Sensing* 47(5), 1318–1327 (2009)

21. Yokoo, M., Suzuki, K.: Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In: AAMAS 2002, pp. 112–119. ACM Press, New York (2002)

Appendix A: Proof of Theorem □

Proof. According to the definition of the encryption and decryption, we have

$$\begin{aligned}
 & \mathcal{E}_+(y; m_1) \odot \mathcal{E}_+(y; m_2) \\
 &= (g^{r_1}, y^{r_1} g_0^{m_1}) \odot (g^{r_1}, y^{r_1} g_0^{m_1}) \\
 &= (g^{r_1+r_2}, y^{r_1+r_2} g_0^{m_1+m_2}) \\
 &= \mathcal{E}_+(y; m_1 + m_2), \\
 & \quad (\text{under new random number } r = r_1 + r_2)
 \end{aligned}$$

while

$$\begin{aligned}
 & \mathcal{D}_+(x; (c_{11}, c_{12})) + \mathcal{D}_+(x; (c_{21}, c_{22})) \\
 &\equiv L_{g_0}(\mathcal{D}_\times(x; (c_{11}, c_{12}))) + L_{g_0}(\mathcal{D}_\times(x; (c_{21}, c_{22}))) \\
 &\equiv L_{g_0}(c_{12}/c_{11}^x) + L_{g_0}(c_{22}/c_{21}^x) \\
 &\equiv L_{g_0}\left(\frac{y^{r_1} \cdot g_0^{m_1}}{(g^{r_1})^x}\right) + L_{g_0}\left(\frac{y^{r_2} \cdot g_0^{m_2}}{(g^{r_2})^x}\right) \\
 &\equiv L_{g_0}\left(\frac{(g^x)^{r_1} \cdot g_0^{m_1}}{(g^{r_1})^x}\right) + L_{g_0}\left(\frac{(g^x)^{r_2} \cdot g_0^{m_2}}{(g^{r_2})^x}\right) \\
 &\equiv L_{g_0}(g_0^{m_1}) + L_{g_0}(g_0^{m_2}) \\
 &\equiv m_1 + m_2 \pmod{p_0 - 1},
 \end{aligned}$$

and

$$\begin{aligned}
 & \mathcal{D}_+(x; (c_{11}, c_{12}) \odot (c_{21}, c_{22})) \\
 &\equiv \mathcal{D}_+(x; (g^{r_1}, y^{r_1} g_0^{m_1}) \odot (g^{r_1}, y^{r_1} g_0^{m_1})) \\
 &\equiv \mathcal{D}_+(x; (g^{r_1+r_2}, y^{r_1+r_2} g_0^{m_1+m_2})) \\
 &\equiv L_{g_0}(\mathcal{D}_\times(x; (g^{r_1+r_2}, y^{r_1+r_2} g_0^{m_1+m_2}))) \\
 &\equiv L_{g_0}\left(\frac{y^{r_1+r_2} g_0^{m_1+m_2}}{(g^{r_1+r_2})^x}\right) \\
 &\equiv L_{g_0}\left(\frac{(g^x)^{r_1+r_2} g_0^{m_1+m_2}}{(g^{r_1+r_2})^x}\right) \\
 &\equiv L_{g_0}(g_0^{m_1+m_2}) \\
 &\equiv m_1 + m_2 \pmod{p_0 - 1}.
 \end{aligned}$$

This concludes the theorem. □

Author Index

- Aref, Mohammadreza 449
Au, Man Ho 80
- Bai, Long 411
Bao, Feng 135, 483
Blanton, Marina 165
- Cai, Shaoying 150
Čapkun, Srdjan 181
Chen, Ping 336
Chen, Shizhan 385
Chen, Yi-Ruei 121
Chen, Zhong 226
Cid, Carlos 32
- Deng, Robert H. 135, 150
Ding, Xiaofei 439
- Eghlidos, Taraneh 449
- Fatemi, Mitra 449
Feng, Dengguo 396
Feng, Zhiyong 385
- Gao, Debin 241
Gao, Feng 411
Ge, Xin-Jing 359
Grabher, Philipp 63
Großschädl, Johann 63
Gu, Liang 226
- Han, Hao 336
Hudelson, William M.P. 165
Hudler, Matthias 63
- Imai, Hideki 474
Inuma, Manabu 474
- Jia, Chunfu 241
Joshi, Ajay 47
- Karame, Ghassan O. 181
Karpovsky, Mark 47
Kawamura, Shinichi 3
Kemmerer, Richard A. 1
- Kiyomoto, Shinsaku 32
Komano, Yuichi 3
Koschuch, Manuel 63
Krüger, Michael 63
Kurihara, Jun 32
- Li, Gen 198
Li, Jianhua 283
Li, Tieyan 150
Li, Xiaohong 385
Li, Yang 3
Li, Yingjiu 150
Liu, Joseph K. 80
Liu, Limin 241
Lu, Kai 198
Lu, Songnian 283
Lu, Xicheng 198
- Ma, Changshe 150
Ma, Chuangui 439
Ma, Tianxiao 459
Malhotra, Prateek 293
Mao, Bing 336
Mao, Wenbo 2
Mathew, Mini 254
Matsuura, Kanta 425
Ming, Jiang 241
Mu, Chunyan 211
Mu, Yi 91
- Niu, Panpan 459
Noguchi, Masatoshi 474
- Ohta, Kazuo 3
Olsson, Tomas 308
- Page, Dan 63
Pan, Yun 493
- Rao, Guozheng 385
Ruan, Anbang 226
- Sakiyama, Kazuo 3
Schuba, Christoph 325
Shen, Qingni 226
Shen, Xiaobin 336

- Shi, Lei 226
 Shi, Yi 425
 Shigetomi, Rie 474
 Singh, Gunmeet 346
 Singh, Sarbjeet 346
 Stefan, Deian 293
 Strasser, Mario 181
 Sunar, Berk 47
 Susilo, Willy 80, 91

 Thorncharoensri, Pairat 91
 Tian, Yuan 370
 Tzeng, Wen-Guey 121

 Vespa, Lucas 254, 268

 Wang, Li 226
 Wang, Licheng 493
 Wang, Lihua 493
 Wang, Xiangyang 459
 Wang, Xiaoyun 107
 Wang, Yi 336
 Wang, Zhen 47
 Wang, Zhi 241
 Wei, Puwen 107
 Weng, Ning 254, 268
 Wu, Chehai 293
 Wu, Hutong 385
 Wu, Wenling 17

 Xia, Zhengmin 283
 Xie, Li 336
 Xiong, Huijun 293
 Xu, Guangquan 385

 Yang, Yahui 226
 Yang, Yanjiang 135
 Yang, Yixian 493
 Yao, Danfeng 293
 Yin, Xinchun 336
 Yu, Yongxin 385

 Zhai, Zhengde 396
 Zhang, Aixin 283
 Zhang, Fuzhi 411
 Zhang, Hao 370
 Zhang, Lei 17
 Zhang, Liting 17
 Zhang, Wei 198
 Zhang, Wentao 17
 Zhang, Yan 396
 Zhang, Ying 198
 Zhang, Zonghua 493
 Zheng, Yuliang 107
 Zhou, Jianying 80, 135
 Zhu, Huafei 483
 Zhu, Jian-Ming 359